The USENIX Association Newsletter

;login:

November/December 1993 Vol. 18, No. 6

From the Editor's Desk

I'm writing this from the LISA VII conference which is an unqualified hit – attendance is just under 1,100 people! Who would have guessed that it would almost double again this year. Three tracks; BOFs through the night; pre-SAGE-election fever; wonderful dessert-at-the-aquarium reception. Who could ask for more?

It's the end of the year and time to take stock, as always. We've increased the size of *;login:* and its technical content (though it'd always be nice to have more). We've seen SAGE material become a major part of *;login:* along with fantastic summaries of the standards bodies' work. I'm working on increasing the number of technical articles and commencing the summarizing of periodicals. (Send me mail to join our small list of volunteers.)

Special editorial thanks go to Bigmac (Bryan McDonald) for his timely submissions and editing of SAGE material, and to Nick Stoughton and Jeff Haemer for the standards material. The fine and frequent individual articles by Elizabeth Zwicky, Hal Pomeranz, and Wendy Nather are most appreciated.

And thanks go to our inveterate "bookworm," Peter Salus. How does he devour the bonanza of books he does, whose topics range from the ubiquitous Internet to such esoterica as *The Economist Style Guide*? Kudos go to those who have reviewed new books in our field. And since I'm running out of space here thanks to all contributors. We wouldn't have this newsletter without you.

We have increased our coverage of conferences and symposia in no small part due to the reporters who bring back the news from the front lines. We receive email from folks who cannot attend a conference *thanking* us for your summaries.

I'm looking forward to a new year with a new design (look for it in March or so) and ever increasing quality (and quantity). Please share your articles with us as you write them! RK

The closing date for submissions to the next issue of *;login:* is December 15, 1993.

USENIX

The UNIX and Advanced Computing Systems Professional & Technical Association

General Information

;login: is the official newsletter of the USENIX Association.

;login: (ISSN 1044-6397) Volume 18, Number 6 (November/December 1993) is published bimonthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710. $24 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $50 per year. Second-class postage paid at Berkeley, CA and additional offices. POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Contributions Solicited

You are encouraged to contribute articles, book reviews, and announcements to *;login:*. Send them via email to <*login@usenix.org*> or through the postal system to the Association office. Send SAGE material to <*bigmac@erg.sri.com*>. The Association reserves the right to edit submitted material. Any reproduction of this newsletter in its entirety or in part requires the permission of the Association and the author(s).

Editorial Staff

Membership and Publications

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Telephone: 510/528-8649
FAX: 510/548-5738
Email: <*office@usenix.org*>

;login: is produced with Framemaker 3.1 software provided by Frame Technology and displayed on an NCD X Terminal, donated by Network Computing Devices. The system is served by a Sun SPARCsystem 10 server. Output is sent to a 600 dpi QMS 860 laser printer, donated by Quality Micro Systems. The newsletter is printed on recycled paper.

Conferences & Symposia

Judith F. DesHarnais, Conference Coordinator
USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630
Telephone: 714/588-8649
FAX: 714/588-9706
Email: <*conference@usenix.org*>

Automatic Information Server

To receive information electronically about upcoming USENIX symposia & conferences, finger *info@usenix.org* and you will be directed to the catalog which outlines all available information about USENIX services.

Tutorials

Daniel V. Klein, Tutorial Coordinator
Telephone: 412/421-2332
Email: <*dvk@usenix.org*>

USENIX Supporting Members

ASANTÉ Technologies, Inc.
ANDATACO
Frame Technology, Inc.
Matsushita Electrical Industrial Co., Ltd.
Network Computing Devices, Inc.
OTA Limited Partnership
Quality Micro Systems
Sybase, Inc.
UUNET Technologies, Inc.

SAGE Supporting Member

Enterprise Systems Management

USENIX Association Board of Directors

If a member wishes to communicate directly with the USENIX Board of Directors, he or she may do so by writing to <*board@usenix.org*>.

USENIX Association

Report from Nominating Committee

by the USENIX Nominating Committee

In June, the Nominating Committee began the process of assembling a list of nominees for the Association's 1994 Board of Directors election. We felt our charter was to present the membership with a selection of candidates who could be dedicated and effective members of the Board.

We began with a collection of 37 candidates who had either expressed interest in running, were suggested by current members or USENIX staff, or were thought to be potentially good candidates. While all these people had talents and skills which might be valuable on the Board, we did not think we would be doing our jobs by offering a slate of 37!

To pare down the list, we conducted interviews with the current Board and staff to help us better understand the needs of the Association. When evaluating candidates, we considered their experience, their talents and interests relating to USENIX, their past involvement with the Association, their interpersonal skills and abilities, as well as their potential to provide leadership and vision, now or in the future. In trying to build a slate, we considered what segment of the membership each individual might speak for or represent most clearly, and we tried to achieve a representative balance on the Board. During the process, our list grew and shrank as more names came up, more people expressed interest, and some of those being considered indicated they would be unable to run.

After dozens of phone calls, electronic mail messages, and hours of deliberation, we selected a list of candidates. These candidates have all shown their commitment to USENIX by participating extensively, including serving on the current Board, running workshops or conferences, teaching tutorials, or serving on committees. In many cases, the new individuals have attended Board meetings, often at their own expense, to educate themselves on how the association is run. USENIX is truly an international organization now, with over 22% of the membership non-US. You will find several candidates in the slate who represent areas outside the U.S.

Regrettably, not all the current Board members appear on this slate. In most cases, these valiant souls have found that their professional responsibilities now require more time and attention than would allow them to continue on the Board. We wish them well and thank them for their excellent service to the Association.

In the past, the officer candidates have generally been chosen from amongst the current Board, people who have had enough experience with the Board and organization to assume the legal, financial, and leadership responsibilities necessary as officers. We chose to follow this precedent. We decided to nominate only one candidate for each of the officer positions to avoid losing experienced and capable Board members.

We offer this slate as a collection of well-qualified, dedicated USENIX members who could provide good leadership on our Board. At the same time, there are certainly many other qualified members who could be running. If you feel the slate is insufficient and would like to run yourself, or know someone who wants to run, nominating by petition is trivial (see pg. 10 of this issue).

It is our pleasure to present the following candidates for potential election to the USENIX Association Board of Directors.

Board members at large (4 slots available):

President: Steve Johnson, *Melismatic Software*
Vice President: Eric Allman, *University of California, Berkeley*
Secretary: Lori Grob, *Chorus systemes*
Treasurer: Rick Adams, *UUNET Technologies*

Board members at large (4 slots available):

Matt Bishop, *University of California, Davis*
Peter Collinson, *Hillside Systems*
Daniel E. Geer, *OpenVision Technologies*
Andrew Hume, *AT&T Bell Laboratories*
Sue LoVerso, *Thinking Machines Inc.*
Greg Rose, *RoSecure Software Pty Ltd.*
Henry Spencer, *University of Toronto*

Submitted by the USENIX Nominating Committee:

Deborah Scherrer, Chair, *mt Xinu*
Jaap Akkerhuis, *AT&T Bell Labs*
Neil Groundwater, *SunSoft*
Pat Parseghian, *AT&T Bell Labs*
Margo Seltzer, *Harvard University*

UNIX Security Symposium Report

by Calum D. McKay
<calum.mackay@mrc-bsu.cam.ac.uk>

The Fourth UNIX Security Symposium was held on October 4-6, 1993 at Santa Clara, California.

Keynote Speech

The keynote speech was given by Robert Morris of the National Security Agency who started by generally defining the role of the NSA as 80% eavesdropping on foreign communications and 20% protecting US communications.

Morris' main theme was that "software produced in the US is largely crap," being badly designed and badly implemented. He pointed out that there are bugs in UNIX versions being delivered now that he was responsible for introducing 18 years ago.

Interestingly he noted that already a noticeable amount of the UK trade balance is made up of imported software. Although this is not yet a problem in the US, he feels that it will not be long before Japanese imported software becomes a problem, just has it has with the car industry. He believes that the US software industry cannot prevent this happening in its current state.

Specifically in the area of problems with UNIX, Morris focused on the wild proliferation of *setuid root* programs, many of which do not need this privilege. UNIX is also much too large nowadays. It started off as the smallest time-sharing system and is now rife with duplication of utilities. This continues through a tendency not to throw away old redundant utilities. In addition, there is a tendency to put far too much into a program, making it far too big, and thus far more liable to fail, or to become a security problem.

The inherent design of UNIX means that it requires a very good system administrator to make it really secure (although it can be done). A mediocre system administrator will just not be able to keep it secure. This design requirement to have good sysadmins, without which UNIX just falls apart, is one of its major problems, and one that needs to be cured. Here Morris made much of his main point, which is that if a system is full of bugs, what can you say about its security?

He has done work for the UK government that has resulted in the publication of strict requirements for software-based systems in safety-critical areas, and he noted that this needed to be introduced in the US also, and a book on standards of good programming practice should be written.

In the "olden days" too much emphasis was placed on program content with too little attention to the user interface. This began to be addressed in the 1980s, and now things appear to have gone too far the other way, with a great deal of time being spent on user interface and almost no attention being paid to program content. Things need to be taken back a bit and a sensible balance needs to be found.

The importance of documentation writing was stressed, as was the need for proper software testing, although testing alone is not enough. The formal methods system was originally a good idea but has been hijacked and messed up by the security agencies. Generally the methods are applied to something other than the code itself, i.e., the comments or specifications, and this is not a good thing.

Morris feels that we should be learning from experience, yet this is clearly not happening, with project teams being broken up after a project comes to an end, and the collective experience is thus lost.

Another problem is the software companies' unwillingness to fix bugs. In general not only are bugs not fixed when the companies are notified, they are often never fixed, frequently being carried over to new releases. This is becoming a "big thing"; recently there was a ten-page article in the *New York Times* detailing Microsoft's response to bugs in Word for Windows, which apparently is to do nothing about them.

In the question session following the talk it was suggested that many of these problems are due to market forces, i.e., users want the new release of a product out on the streets as soon as possible and the software companies are forced to compromise quality in order to get the product out ahead of their competitors. Morris agreed with this statement.

CryptoLib: John Lacy

Cryptography traditionally has been hardware-based, but to do research in the field requires the

flexibility that only software can provide. Can software do the job efficiently? Software has always been viewed as too slow, but with today's processors this is no longer true.

New applications are in need of cryptography all the time, e.g., interactive entertainment, electronic commerce, virtual meetings, etc. Various types of key encryption authentication systems are showing their age; Lacy mentioned a recent paper detailing a theoretical machine costing about $1M which could search the entire DES space in about 7 hours. Most public-key systems are feasible when implemented in software but are more suited for applications like digital signatures rather than bulk data encryption. Random number generation is very important for key generation, and there is a lot of effort being devoted to improving these, and even "true" random number generators based on things like system interrupts and clock skew. There has been some success here but it is slow. The main application seems to be to seed traditional, pseudo-random number generators which are used for faster key generation. There was some discussion of prime random-number generation.

Details were given of a library of tools the authors built to facilitate research into key-based systems. Future research involves building a protocol front end to their library, and finding ways to make cryptography more widely used and included automatically in all new applications.

Long Running Jobs: Aviel Rubin and Peter Honeyman

The authors run a large distributed system in which Kerberos provides an authenticated environment. A problem occurs with the ticket-based mechanism that Kerberos uses, in that the ticket is only valid for a day or so. If a job is to run for three weeks it will need to re-authenticate itself to get a new ticket every day, and this requires the user's password to be hard-coded into the program. Their solution is to use a replacement for the *at* program called *khat*, which allows re-authentication using a ticket-granting ticket mechanism. The system seems to work well, and is freely available.

Network Layer Security: John Ioannidis

Application-level security is just not sufficient nowadays. A general network-layer security mechanism is needed, and to that end the authors have developed a platform called *swIPe* for researching security policy management. This system gives authentication, data integrity (by a method based on detection not prevention) and data confidentiality. All this is available to data in transport only.

This system can be used between two machines end-to-end and also in router implementations to secure access to networks. It is implemented by optional encryption, and then encapsulation of IP datagrams which are then sent out at the network layer as a separate IP protocol type. It should work under any BSD-based system using the virtual network interface. DES & MD5 are used for encryption and authentication respectively, and the process adds 30–50 bytes per packet plus padding.

SecureBase: Jon Kamens

The author addressed the problem of how to deal with third-party binaries that come supplied without adequate security. Many of these send passwords and data across the network in the clear, and it was demonstrated how, with 21 lines of Perl script and the *etherfind* utility, one can discover Sybase passwords. A long-term solution to this would be a complete security environment. The short-term solution is to try and make the application safer yourself. There are various ways to do this; altering the application source code is one option but is difficult and expensive. If the application comes with security hooks these can be used but they are often not sufficient. Some sort of wrapper mechanism appears to be the best solution, either within the application, in client programs, or below the application at the stream level.

The author chose a combination of these methods – Kerberos APIs to link in and a wrapper at the Sybase protocol layer. Future work on the project involves adding encryption, improving the performance, generalizing and applying to other third party applications. Their code is not yet freely available.

Packet-based Filters: Jon Boone

The author is researching the performance implications of using fire walls. For most people the firewall machine will be faster than the line connection so no problem exists. However, as communication lines get faster, problems might begin to arise. Using a DEC Alpha for routing, it is possible to obtain 60 Mb/sec on an FDDI ring, but add a packet filtering program like *screend* and it drops to 30 Mb/sec, and packet processing times jump from 400 Msec to 600 Msec. This talk, which only lasted five minutes, was quite disappointing given the relevance of the subject matter. In addition, no paper was presented.

Dial-in security: Bob Baldwin

This talk highlighted what the author referred to as the modem problem. Namely that modems are cheap, fast and seem ideal but are a big security problem. A single protected modem point of entry is relatively secure, but in a big organization this does not work; modems appear everywhere in the company, almost by themselves. All these modem entry points must be protected, yet without handicapping access. Various solutions to this problem are dial-back modems, dial-back software, and one-time passwords.

The author claimed that 1) ordinary dial-back modems can be defeated, although password-based dial-back would seem a way around this, and 2) that one-time password tokens are a bad solution since people forget to take the card with them or the batteries die. The author's company markets the other solution, being a system based on dial-back software. One strong advantage of this method is that currently installed modems can be used.

The software appeared to have nice accounting and billing features. The point was made that, as with all security software, its own setup should be continually audited by comparing with a known secure version.

Secure RPC: Dave Safford

Following a serious break-in, it was noticed that crackers were making frequent use of reading clear-text passwords from the network. This suggests that some form of encryption is required. A short term solution that could be easily implemented was aimed at, one that avoided patent problems by writing on top of the already existing secure RPC implementation. An important criteria was that it be compatible with existing methods such as Kerberos and SPX. The method used seems to work well, but has two disadvantages; it only authenticates the client to the server, not vice versa, and the secure RPC algorithm itself is subject to logarithm-based attacks, although this currently requires considerable CPU time.

Caller ID System: Ghun Choe

The authors were concerned with how to trace and get information back from each node traversed by a remotely-logged-in user. Yet again following recent break-ins, they determined to trace the route back to the crackers. Originally the *tcp_wrapper* program was used, but it was found to be difficult to trace all the way back to the originating host. The authors' system works by keeping track of the path used and this is verified at each stage. Tests show this system works, and the design ensures that at least one compromised host in the path can be detected. One drawback is that the software must be installed on each machine, so PCs and terminal servers cannot be included in the scheme.

ATP: David Vincenzetti

ATP is basically a file integrity checker similar to Tripwire. The wide diversity of attack methods available to crackers make it almost impossible to ensure that a system is ever really secure. It is quite easy for a highly skilled cracker to break in to an Internet connected site. The cracker can then create a back door for himself and retreat without leaving any trace of his intrusion. This means it can be difficult to tell when an initial break-in has occurred, and thus makes it very important to be able to detect and subsequently analyze such attacks.

ATP does this by gathering information about files and storing it in a database file which can be checked against later. This database is protected with CRC and MD5 checksums and access to it is controlled by DES encryption. ATP can them perform various audits on the filesystems and report changes to any files.

Advantages over other similar systems are its configurability and the security of the database itself. The system is freely available.

TAMU: Dave Safford, Douglas Schales, & David Hess

As seems very common, a serious break-in prompted the development of these tools. After watching intruders for a few days, it became clear that a filtering implementation was required at this site since many of the campus machines were not under their control, and thus could not be made secure. Initially, deny-based filtering was used (i.e., allow anything not specifically denied). This simple implementation was too insecure and quickly broken through, so it was changed to an allow-based filter set (deny unless specifically allowed). The crackers were found to be very well-prepared with operating system sources, programs to try many known holes, fake checksums, etc.

Further work involved improving filter flexibility, as well as writing monitoring tools, and a host checking tool.

Filtering is done on a 486 PC with two Ethernet cards running a filtering program like Drawbridge. This approach was used since the standard router filter sets were found to be not flexible enough. On the Internet side of this PC is a workstation monitoring all traffic. This is required since the general approach is not as

secure as a bastion host system, which would be too restrictive. Since some things are bound to get through, this monitoring function becomes really important. It is situated on the outside of the filter so that it can log rejected packets also. The main purpose of the logging is to record network events and spot unusual activities. The logging programs collect about 4MB of data per day, then an extraction program takes out all the expected traffic flows to see what is left. A separate program analyzes the application level protocols and looks for unusual activity. A statistical tool also looks for unusual mixes of protocol and port usage.

Future work includes porting to other machines, an expert-system based log analyzer, and logging of other protocols. The package is currently available for Sun systems.

The second part of the authors' system is the "Tiger" script. This is a checking program along the lines of Cops. In writing this, the main goals were to make it easy to use, with clear explanations of problems found and how to fix them. Also of importance was that the program be thorough and easy to port. It has advantages over Cops in that it is easier to use and incorporates more checks.

The system includes a secure digital signature checker based on MD5 and comes with a database of the correct signatures for the operating system distribution, including patches. Like Cops, once it is setup it can be run from *cron*, reporting only the changes since the last run.

The authors plan to make updated signature databases available via *ftp*.

UNIX Security Update: Jerry Carlin

The author presented a summary of points noticed during several years of dealing with UNIX security, the main points being: there are crackers out there everywhere; there will always be security bugs in the operating system; it is difficult to ensure that you not miss something critical; and it is important to use technology to fight human nature and laziness. The author recommendation was: "Don't worry" (although he admitted that this might be difficult).

The Persistent Hacker: Eduardo Rodriguez

After 7 years of assorted UUCP, X25, & BITNET access, the author's site gained Internet access. Almost immediately a cracker broke into their system by using trivial passwords and gained root. The cracker then patched *telnetd* and */bin/login* to give him further passwords and then used the system as a base to attack other systems around the Internet.

The author concluded by noting that it is difficult to keep as up-to-date as the crackers with regard to security problems, and it is difficult to balance security against ease of use.

Sendmail Without the Super User: Mark Carson

The author stressed the importance of the least privilege principle, whereby a program is given no more than the required privileges to accomplish a task, and it drops these when they are no longer required. It was noted that "secure" software will always have bugs, especially if it is large, and by limiting the scope of the privileges given to this program one can limit the effects of these flaws.

Sendmail is a good example of this problem and it was decided to try and improve matters. The first task was to alter the *sendmail* system to run *setgid mail* rather than *setuid root*. For the few remaining tasks requiring root access the necessary privileges were acquired only when they were needed and dropped immediately afterwards.

In all, the code changes amounted to no more than 500 lines out of 26,000.

Clark-Wilson Access Triples: Tim Polk

Although many have pointed out the dangers of *setuid root* programs, this talk presented the idea that in some circumstances they can be used to enforce your security policy.

A 1987 paper by Clark & Wilson introduced the idea of enforcing an access control triple {*X*,*Y*,*Z*}, whereby user *X* can modify data *Y* only by using procedure *Z*. This sort of type enforcement is not really available in UNIX, but the *setuid* mechanism is similar to type enforcement domain transitions, and in simple examples can be used to enforce an access control triple (although it breaks down in more complicated examples). Unfortunately, a late-discovered error in this work severely limits the implementation and several other limitations exist anyway; but, with work, the *setuid* mechanism can be used to enforce some access control policies.

It was concluded that UNIX would benefit from a better way to enforce these ideas.

Works-in-progress Sessions

Ed DeHart from CERT presented some thoughts and recommendations. It was noted that sites spend a lot of time securing themselves and then let users log in from insecure sites. Crackers gain access to one machine, run programs such as *tcpdump*, and watch for passwords and other useful

data that will allow them to exploit further. It was strongly recommended that a filtering router/fire-wall solution be used to avoid protocol attacks based on NFS/NIS, etc.

Fred Avolio from TIS detailed some recent work connecting the Executive Office of the President to the Internet (*whitehouse.gov*). There was some interest expressed as to whether the President has yet expressed a preference for *xmosiac*.

Bill Cheswick from AT&T talked about the smart card system that he hopes AT&T will make available soon. He freely admitted that this was a blatant marketing talk on what amounted to vapor ware but claimed that this was not an unusual event.

Ying-da Lee detailed the recent work on the "socks" proxy software.

Bob Baldwin made a plea for help in decrypting some files that a cracker left behind on his system. The files are encrypted compressed data, and since compressed data is highly structured it ought to be possible to use frequency analysis techniques to decrypt them.

Bill Cheswick closed the symposium by commenting that in his opinion these events could be considered useful if you came away from them with one good idea per day. On that basis I can declare the Fourth USENIX Security Symposium a resounding success.

Board Meeting Summary

by Ellie Young
<ellie@usenix.org>

Below is a summary of some of the actions taken at the regular quarterly meeting of the USENIX Board of Directors convened in Santa Clara, California on October 3-4, 1993.

Attendance:

Rick Adams, Eric Allman, Tom Christiansen, Lori Grob, Steve Johnson, Evi Nemeth, Mike O'Dell, Barry Shein, Ellie Young, Judy DesHarnais, Diane DeMartini, Dan Appelman

Budget

Young explained that a modest net had been budgeted for 1993, but that this could change depending on attendance at the LISA and Security events. Adams remarked that the lower attendance at the general conferences had significantly effected the budgeted net figure. It was suggested that a capital expense fund be created. The preliminary draft budget for 1994 was discussed in depth, so that a final one could be prepared for the coming year. It was suggested that an item for advertising revenue for *;login:* be added, and that we actively solicit advertising for the newsletter. Funding to have Jeff Haemer and Nick Stoughton continue as the Standards Liaison and Snitch editor respectively was approved. It was decided to raise fees for the 1994 LISA tutorials by $30 to $320 for one-day; $590 for 2 days. It was decided to accept the MIT Press's recommendation to increase non-member subscriptions to the journal to $65.00.

4.4 BSD Manuals

Young reported that the 4.4BSD manuals were ready to be published, once the lawsuit makes it free and clear to do so. After some discussion on the potential market, it was decided that since USENIX had already invested a significant amount of funds to make them POSIX-compliant, that even if the market was small we should produce them as a service to the community. It was agreed to increase the budget line item for the 4.4 BSD Manuals, and to form a committee of O'Dell, Allman, and Young to oversee the production of 1,000 copies of the manual with accompanying CD, possibly containing 4.4BSD-lite, the documentation, and a CD.

CDROM

Christiansen and Grob formed a subcommittee to look into having USENIX producing a CDROM.

Executive Director's Report

A member/attendee survey would be conducted in November/December. Young solicited the board's input on *;login:*'s content. It was felt that there had been an overall improvement in the newsletter, and in particular the contributions from SAGE. It was also agreed to extend the program where publishers offer books to members at a discount to include other products, as long as all vendors were treated equally. It was also suggested that there should be a page in *;login:* to summarize member benefits.

Mobile Computing Symposium

Those who attended reported that the symposium was successful and the keynotes were well received. Young reported that she was discussing the possibility of plans for another one that might be co-sponsored with the IEEE and/or ACM.

Microkernels/SEDMS

O'Dell said there was a lot of discussion about our desire to combine into one meeting the Mach/Microkernel/SEDMS programs and the consensus among the organizers is that we should do this. There was a lot of discussion about keeping the "experiences" an important part of its focus and how to position it against SOSP. Grob felt that the program chair (Jay Lepreau) would be able to address the issue of serving the different communities.

LISA 1994

Young reported that the subcommittee had decided unanimously to accept Dinah McNutt's proposal to serve as program chair for the 1994 LISA which would be held at the Town & Country Hotel in San Diego, September 19-23, 1994.

Affiliates and Locals

Johnson summarized the various reasons why local groups might wish to affiliate with USENIX and/or SAGE, i.e., insurance, discounts, mailing lists, use of name, and member services. Appelman cautioned that in order to ensure our tax-exempt status, it is the Association's responsibility to be sure that the affiliate's purpose is consistent with the USENIX charter, and that their activities are reviewed on a regular basis. Young stressed that our member benefits should not be discountable, and that we not offer the same benefits to groups that are not USENIX members. In order to encourage international groups participation, however, it was agreed that we continue to extend discounts on conference registration fees to members of EurOpen's national groups, JUS, and AUUG. It was also suggested that when/if we consider doing a joint membership with local affiliates, that it be 90% of combined dues.

Relations with Other Groups

Johnson said he had not yet made a trip to liaise with EurOpen national groups, but that either he or Young would do so in November. Young explained that at this date it was unlikely that UniForum would be able to co-locate its 1995 conference with us in a January timeframe, and she would continue to discuss future possibilities.

Awards

O'Dell reported that the committee is very close to making a decision on the recipient for the 1994 Flame award (which recognizes overall achievements and outstanding contributions to the community). He and Young had been in touch with Scherrer about our administering an award from the funds leftover from the now-defunct Software Tools User Groups. The STUG award would recognize a unique or interesting development in software tools.

Regional LISA Seminar Idea

It was decided to defer the discussion of whether we should consider offering regional LISA seminars/tutorials to the SAGE board.

1994 Elections for Board of Directors

by Ellie Young
<ellie@usenix.org>

The biennial elections of the Association will be held in the Spring of 1994.

A report from the Nominating Committee appears on pg. 9 of this issue. Nominations from the membership are open until January 28, 1994. The procedure for nominations by the membership is a written statement of nomination signed by at least five (5) members in good standing (or five separate nominations), to be submitted to the Executive Director at the Association office, and received by noon, PST, January 28, 1994. Please include a Candidate's Statement and photograph for inclusion with the ballots as well.

All nominees are invited to attend the Candidates' Forum to be held alongside the Open Meeting of the USENIX Board of Directors, sometime during the week of the Winter 1994 Technical conference, January 17-19, 1994, at the San Francisco Hilton Hotel (watch *comp.org.usenix* for time and location). Members are urged to come to this meeting with their questions!

Ballots will be sent to all paid-up members as of February 23, on or about February 28. Members will have until March 31 to return their ballots, in the envelopes provided, to the Association office. The results of the election will be announced in *comp.org.usenix*, at the Summer '94 General Conference in Boston, and in the May/June issue of *login:*.

The Board is made up of eight directors, four of whom are "at large." The others are the President, Vice President, Secretary, and Treasurer. The balloting is preferential, with those candidates with the largest number of votes being elected. Newly elected directors will take office immediately following the conclusion of the Annual Meeting of the Association which is held in June at the Boston Conference.

USENIX Member Benefits

As a member of the USENIX Association, you receive the following benefits:

- Free subscription to *;login:* — technical features, system administration tips and techniques, international calendar of events, SAGE News, media reviews, Snitch Reports from the USENIX representative and others on various ANSI, IEEE, and ISO standards efforts, and much more.
- Free subscription to *Computing Systems*, refereed technical quarterly published with The MIT Press.
- Discounts on registration fees for the large, multi-topic Winter and Summer technical conferences, LISA, the System Administration conference, the C++ conference, and the various single-topic symposia addressing topics such as UNIX Applications Development, Security, Operating Systems, High-Speed Networking, and Mobile Computing – as many as ten technical meetings every year.
- Discounts on proceedings from USENIX conferences and symposia and other technical publications.
- Discounts on the USENIX Association book series published by The MIT Press. Now available: *The Evolution of C++: Language Design in the Marketplace of Ideas*, edited by Jim Waldo of Sun Microsystems Laboratories.
- Savings (10-20%) on selected publications from Addison-Wesley, McGraw-Hill, The MIT Press, Prentice Hall, John Wiley & Sons, O'Reilly and Associates, and UniForum.
- Right to join SAGE.

Contact the USENIX Association office at 510/528-8649 or via e-mail *<office@usenix.org>* for help placing publications orders.

Vendor Exhibitions – Should you Exhibit?

by Peter Mui
<display@usenix.org>

Vendor Exhibitions are held at selected USENIX conferences to showcase products we feel are of interest to our membership. We try to have unique exhibitors that you might not be able to see at another show because of their specialized appeal or the size of the company. Some criteria we like to use:

- Regional companies geographically local to our conference site
- Smaller companies, but with products otherwise of interest
- Divisions of corporations i.e., technology-driven divisions of large companies with a specific interesting product to display

Many USENIX members may work in environments that fit these criteria. Should you or your company be exhibiting at USENIX Conferences? We try to make it as easy as possible for a small company, or a division of a large company, to exhibit with none of the hassles associated with going to a trade show and at a fraction of the cost. The exhibitor's fee is scaled to the number of expected attendees, and includes all of the furniture and drapery for tabletop presentation, which is set up in advance.

In 1994, we will be having vendor exhibitions at the following conferences:

Winter Conference, January 17–21
San Francisco, CA (2000 attendees)

C++ Conference, April 11–14
Cambridge, MA (300 attendees)

Summer Conference, June 6–10
Boston, MA (1500 attendees)

LISA Conference, September 19–23
San Diego, CA (1000 attendees)

For more information on exhibiting at these shows, contact Peter Mui at the USENIX office: 510/528-8649 (phone), 510/548-5738(FAX), or send email to *<display@usenix.org>*.

Community News

Invitation from David Cannon *<D.Cannon@exeter.ac.uk>* "A number of IEEE POSIX drafts are scheduled for input to ISO/IEC JTC1/SC22/WG15 over the next year. If you are based in the UK, and if you are interested in assisting the progress of your standard through ISO, you are qualified to attend and take part in the UK group which feeds into ISO WG15. This group is more formally known as the British Standards Institute (BSI) POSIX panel, IST/5/-/15.

There are no fees associated with membership of IST/5/-/15. You qualify simply by living in the UK and having an interest in the work. I look forward to hearing from you."

David H. Wolfskill *<david@dw68k.cts.com>* reports: "After more than 10 years as an IBM mainframe systems programmer, I'm finally making the move: as of November 1, I'll be joining Bruce Robertson at Hundred Acre Consulting, Reno, Nevada, to help support publicly redistributable software. I will be reachable as *<david@owl.pooh.com>*; *<david@dhw68k.cts.com>* should continue to work for my home machine."

Annadiana Beaver *<abeaver@cea.berkeley.edu>* announces her marriage to Massoud Abedini on October 29th, 1993. She will be changing her name to Annadiana Abedini.

J.R. Oldroyd *<jr@opal.com>* is pleased to announce the formation of his new business, Open Advisors Limited (Opal). Before founding Opal, J.R. spent ten years at Instruction Set, both in the UK and US.

From the SAGE Board

State of SAGE

by Peg Schafer
<peg@bbn.com>

As SAGE Treasurer, I am writing on behalf of the SAGE Board of Directors to bring you up to date on current actions and achievements of SAGE.

For me personally, SAGE has already made significant contributions in the development of system administration as a profession. For example, a manager in my company was looking for a junior level system administrator, and asked for assistance. I provided him with the excellent job descriptions developed by the SAGE job description working group. After reviewing it, not only did he have a better understanding of the skills required for system administration, but he realized he required a Level III, Intermediate/Advanced System Administrator! The impact of such contributions is real, long term, and verifiable. SAGE is making a difference — not only for the system administrator who will be hired for this position, but for all of us who are "in the swamp!"

SAGE is a very young, active organization. We are cosponsoring with USENIX a number of events designed to improve system management:

- Tutorials on Security and System Administration Management at the UniForum Conference, March 21–22, 1994 in San Francisco

  These tutorials will explore effective management techniques in two important areas: protecting network and system security, as well as providing answers and guidelines for how best to manage and build a team of system administrators.

- With FedUNIX and USENIX, the System Administration, Networking, and Security (SANS) Conference on Tools & Techniques, April 5–9, 1994 in Washington, DC

- The 1994 USENIX LISA conference

Besides conferences and tutorials, SAGE is making significant contributions in the following areas:

- The SAGE Job Descriptions working group has completed the template job descriptions document, and has added supplementary information to create a booklet. *Job Descriptions for System Administrators*, edited by Tina Darmohray, is the first in a series of very practical booklets and resource guides covering system administration issues and techniques, produced by SAGE and available at a discount to SAGE members.

- The Electronic Information Distribution working group has created an archive site, *ftp.sage.usenix.org* for papers from the system administration conferences and system administration related documentation.

- As a result of the discussions surrounding the SAGE Job Descriptions working groups efforts, the *sage-jobs-offered* mailing list was created. This mailing list has been created to provide a central location for external groups to post job advertisements relevant to the SAGE membership.

In addition to the existing member benefits (like conference discounts), USENIX and SAGE now offer:

- a 10%–20% discount with a number of leading publishers of UNIX books

- a 20% discount from the UniForum Association for the annual UniForum Conference

SAGE fever is spreading! SAGE organizations in Australia and the United Kingdom have formed and are developing initiatives of their own. $GROUPNAME in New Jersey has joined the existing regional groups for system administrators, BayLISA in the San Francisco Bay Area and Back Bay LISA in Boston. SAGE is currently attempting to formalize relationships with these groups, and investigate areas for mutual cooperation.

As we go into 1994, SAGE will continue to work with USENIX in fostering technical development and sharing of solutions to problems, as well as communicating with users, management and vendors on system administration topics. We want to encourage and recruit each of you to get involved. Proposals for new services are especially welcome. Let us *<sage-board@usenix.org>* hear from you! And also please vote in the up-coming elections!

Impressions of SAGE-AU '93

by Janet Jackson
<jackson@cwr.uwa.edu.au>

[Editor's Note: This article was first published in AUUGN, the Newsletter of the Australian Group for UNIX and Open System users, Volume 14, Number 4.]

The inaugural annual conference of the System Administrators Guild of Australia came at an interesting time for me: I'd just accepted a new systems management job. I'd already been shown around my new department's systems, so I had two networks to think about as I listened and learned.

And I did learn. I've never before been to a conference where almost every talk presented ideas relevant to my work. And as for having 60 or 70 fellow systems administrators to discuss ideas with... the word that comes to mind is "fabulous."

Wednesday — Tutorial Day

The conference was held at Melbourne University, well placed for restaurants and hotels. My day started with a good breakfast, then a short walk up the street to the conference venue.

Ah, the Solaris tutorial! We waited behind solid wooden desks in the elegant paneled room, knowing its ambience would soon be ruined by the clamor of fifteen or twenty systems administrators grappling with the reality of Solaris 2. The instructors were a bit late; apparently no one told them where on campus to come. This seemed to be the only slip-up in an otherwise smoothly-run conference.

The tutorial was presented by Fraser Gardiner, Rich Baxter and Dennis Letizia, from Sun Microsystems. One of them started by showing us all the screens from Sun's graphical pseudo-sysadmin tool. This was rather pointless, because none of the audience seemed to have any intention of using such an uncustomizable tool, and even if we did, I had no doubt we'd all be bright enough to navigate its simple screens.

Fraser then showed us how to set up Auto-install (otherwise known as Jumpstart), which looks pretty useful. Fraser clearly knows his stuff and gave an excellent practical description — the sort of thing I expect at something called a tutorial. Unfortunately, the other two then presented product pitches for NIS+ (which looks good, but I already knew that) and Sun Net Manager (nothing new there).

After the lunch break it was my turn to present a completely different tutorial ("Perl for Systems Administrators"), to much the same audience. But first, I had to eject the SAGE-AU committee from the room, where they were holding their first face-to-face committee meeting. (Their one previous meeting was held by email, took several hours, and by all accounts was a most successful experiment.)

Three hours later, suffering from presenter's throat, legs and eyes, I hated to think how Brent Chapman, who had presented the all-day Internet Firewalls tutorial to a packed room, must be feeling. (But don't let me put you off speaking at conferences! The rewards outweigh the inconveniences.)

Thursday — Conference Day 1

After a welcome from SAGE-AU President Hal Miller, Elizabeth Zwicky shared her experiences with heterogeneity: trying to integrate different machines and operating systems into a coherent network. She distinguished coarse, medium, and fine heterogeneity. *Coarse* meant completely different operating systems, like UNIX and Macintosh, where you want to provide common services, such as email. Elizabeth's advice here was to use the simplest possible protocols. *Medium* meant different vendors' implementations of the same operating system. Elizabeth spent most of the talk describing the specifics of integrating different implementations of UNIX. *Fine* heterogeneity, the worst kind, meant different implementations (of UNIX) from the same vendor (Solaris 2 and SunOS 4, for example).

After coffee, Gregory Bond described how his company made a successful decision to put X-terminals, rather than workstations, on their users' desks. He gave us a lot of information about the pros and cons of both solutions.

Mike Selig, of Functional Software, described the rather elegant database approach used in Functional's industrial-strength UNIX administration software. I've heard a number of talks about this work, and I thought Mike explained their approach particularly well.

We then enjoyed a stand-up buffet lunch in the combined exhibition/registration area. The food was nice, but anarchistic queueing caused contention, with people trying to move in several different directions.

The exhibition was low-key, with stands from a couple of vendors and a bookseller.

Labtam and Melbourne University provided X terminals and an Internet connection. There was some trouble getting them going, despite at least ten well-meaning advisors and about fifty others ardently ignoring the problem (as only sysadmins know how).

After lunch Geoff Huston, an impressive, entertaining speaker, explained the status of AARNet – traffic doubling every eight months! – and attempted to predict its next few years. He thought it could become too big for the current owners (the Australian Vice Chancellors' Committee) and may end up privately owned.

Neil Brown from the University of New South Wales told us that "Installing Software can be Fun." Despite the title, I didn't enjoy this talk. I found it frustrating. Neil's team controls their software installation with his Conform system and by programming in a rather esoteric special-purpose language (or perhaps the code is automatically generated; I am not sure). This idea sounded interesting, but Neil spent a lot of time describing the language, in tedious detail, without giving a clear description of the overall system. It was also unclear how well the system worked in practice.

After a coffee break, SAGE-AU held its first Annual General Meeting, an informal affair. Having done such a good job of the conference, the committee was re-elected.

Conference Dinner, Thursday Evening

It seems that by and large, systems administrators are a scruffy lot. Nevertheless, they scrubbed up reasonably well for the conference dinner, which was held in Melbourne University's rather sumptuous University House.

The food was good (although fighting through a pastry crust to drink my soup is not really my thing), the conversation stimulating and the wine plentiful. I was amused to discover that I am not "a reasonable female," because I don't like port.

Friday – Conference Day 2

Like many people, I was late for Mark Andrews' talk on how to clean up a Domain Name Service hierarchy. (This was partly an after-effect of the dinner, and partly because the talks started half an hour earlier than the previous day's.) Mark's was a tutorial-style talk explaining various mistakes people commonly make with their DNS records. He gave lots of information.

Brent Chapman then woke us up with a short, animated talk on the do's and don'ts of Internet Protocol packet filtering. An important point was that different vendors' routers will apply your filtering rules in different order, which can make a big difference to which packets they let through.

Next, Gregory Bond explained how to convert an old diskless workstation into an X-terminal. This talk was an excellent short tutorial; maybe that's what I can do with that old Sun 3 in my new department.

I spent the coffee break getting increasingly nervous, because I was the next speaker. I was surprised to find I was so much more nervous about a fifteen-minute talk than about yesterday's three-hour tutorial: I'll blame the preceding speakers for setting such a high standard!

I was disappointed to be giving the only fifteen-minute work-in-progress talk: I hope to see more of these at future conferences. I described a way to distribute certain networked system files, such as *printcaps*, using a recursive invocation of *make*, and was pleased with the number of ideas and suggestions that came from the audience. Sharing ideas is one of the things SAGE is for.

I felt much better all of a sudden, and enjoyed David Jones's description of his university course in systems administration. He takes a small class for a semester, giving them a UNIX system to look after. His course involves gradually teaching the students the basics of systems administration, from software installation to ethics, and includes an "emergency week" where students are expected to cope with simulated crises. The course doesn't make them good systems administrators (not in 13 weeks!) but does give them a head start in learning how to be good systems administrators. David mentioned one problem with the course: no one seems to want to use the student machine for anything much, so it's hard for the students to get experience with real users. Another problem I can see is that academics interested in systems administration appear to be pretty rare.

Peter Billam then described the environment he provides for his users at the Tasmanian Treasury Department. He has developed a simple, rather elegant login shell that presents users with a menu of options, including "Change preferences," "Contact systems administrator," and "About," as well as mail, news and site-specific

applications. A pleasant surprise was that this is all done under 386BSD.

The first afternoon speaker was Elizabeth Zwicky, who talked about backups for an hour and a quarter. I thought this was too long, for both speaker and audience. It was really two talks in one: the first, about how to design a good backup policy, and the second, about the inadequacies of the various UNIX backup programs. Having already read her excellent LISA conference paper on the second topic, I found the first topic much more interesting, but other attendees may not agree with this.

Steve Landers then presented a different angle on the same topic, with his description of Functional Software's backup scheduling architecture. This system provides a coherent, flexible front-end through which a relatively inexperienced operator can drive any of the UNIX backup programs. (Steve seems to like *cpio*, whereas Elizabeth prefers *dump*.)

After coffee, Srdjan Nikolic described the Remote Console Access Facility (RCAF) developed by Fujitsu for a client. I was pleasantly surprised that this was definitely not a typical "vendor" talk. Srdjan gave a good deal of technical detail about the RCAF, which provides logging and network management as well as a system console located at the other end of a networked or dedicated connection.

The final talk was about Silicon Graphics' NetVisualyzer. This did seem to be a "vendor" talk, extolling the virtues of the product without telling us very much about it (although a few screens of it were shown). However, it was presented by a customer, Peter Bonnello from BHP Minerals International. He had been using NetVisualyzer for a short time — not really long enough to give a balanced view of it. Although vaguely interesting, this talk was not strong enough to be a good conference-closer.

After the closing remarks, a bunch of us adjourned to the nearby Clyde Hotel. Here, I ran into an amazingly familiar-seeming man who certainly knew me. However, after three days conferencing, my brain was overloaded with new faces, and I just couldn't think who he was.

"What are you doing here?" he asked.

"Drinking with some people over by the window."

"But what brings you to Melbourne?"

"I'm here for the Systems Administrators Guild conference."

At this point I still hadn't placed him, so I excused myself and returned to my seat. After about five minutes, some background job in my mind finally exited, and I suddenly remembered him. He was one of the people who had interviewed me for my new job!

A few days later I saw him again, apologized, and all was well.

Security Tool Review: TCP Wrappers

by Mark A. Monroe
<monroe@mercator.rtpnc.epa.gov>

[SAGE editor's note: Chistine Quinn of Stanford University has been diligently working to create a regular column to review software tools for systems administrators. I am pleased to present the first of many such reviews. Mark Monroe and Dan Farmer have teamed up to review security tools for SAGE, and their first offering is a good one. If there are tools that you think Christine should try to have reviewed in future columns, please contact her at <quinn@ee-cf.stanford.edu>.]

"May I see some identification, please?"

The security guard asks this question of each person entering the lobby of our building, and the presence of the guard keeps all but the most determined intruders outside the foyer.

If you imagine that your computer is an office building, the TCP wrapper program becomes a security guard at the entry desk, recording the comings and goings of everyone who accesses the "building." In addition, the wrapper acts like a security gate, checking IDs before allowing entry. Wrappers are not foolproof, but they do go a long way towards closing the door on unwanted visitors.

The wrapper program, add-on software written by Wietse Venema, does not require changes to existing daemons or configuration files. It does not exchange data with the client program, which makes it invisible to the authorized user and

insensitive to bad data from the requestor. It is also independent of application (*telnet, rlogin* etc), so the same protection can be used to cover many services. It is available via anonymous *ftp*. (See below.)

How Does It Work?

TCP wrapper checks each request your computer receives for TCP/IP services like *ftp, telnet,* Berkeley *r* commands, etc. It compares the source address of incoming requests to a list of hosts that are allowed or denied access to services on your machine. Every request, successful or not, is logged by *syslog* so you can see who is requesting access to your computer.

The concept is simple: *inetd* is configured to call TCP wrapper in the place of the daemon for each service you wrap. This gives the wrapper a chance to check two files, *hosts.allow* and *hosts.deny,* for the hostname or Internet address of the incoming request. The allow/deny files are queried each time a connection is requested, so changes to the files are picked up without a SIGHUP to an existing process.

If the *TCP* wrapper can match the incoming request with a line in the *hosts.allow* file, the request is logged, the connection is passed to the appropriate daemon program and the session continues as if there were no wrapper. If the wrapper finds a match in the *hosts.deny* file, the request is logged and the connection is dropped. Keywords (e.g., ALL) and wildcards in the allow and deny files make configuration easy. The entry ".edu" refers to all connections from host addresses ending in that string.

TCP wrappers can be configured to compare the address inside the IP packet to the host name that is returned when the remote site is queried with `gethostbyaddr()`. The wrapper disconnects if there is any discrepancy, suspecting that the requestor is trying to spoof your machine.

You can even do counterintelligence of a sort. The allow and deny files can be configured so that the remote site is identified using a reverse *finger* (or any arbitrary command) to try and identify the real user making the request. The RFC 931 protocol is also supported by TCP wrappers.

The README file contained in the TCP wrapper package is excellent and has a more thorough explanation of these and other features.

How Long Does It Take To Install?

I retrieved the *shar* files and had the program built and installed two hours later on my Data General AViiON. Most of the time was used reading the documentation, changing configuration files, etc. The build takes only a couple of minutes. I was able to build and install on a Sun 4c machine in about thirty minutes, start to finish. We tested four other machines and had similar results. The README file lists 12 machines that have been tested successfully, including SunOS, DEC Ultrix and OSF/1, HP-UX, AIX, Apollo, Sony, NeXT, SCO UNIX, DG/UX, and UNICOS on a Cray.

Shortcomings

Version 5.1 of TCP wrappers doesn't support the System V TLI protocol yet. I tried an alpha test version that had been modified to accept TLI connections, and expect to be able to wrap TLI on DG/UX and Solaris very soon.

In the README file, other shortcomings are discussed, including the difficulty in detecting host name and host address spoofing. The current version of TCP wrappers tries to detect host name spoofs by checking the name-address combination from two sources. Wrappers also only check the initial connection request, making it unusable on NFS mount daemons or daemons that linger after the initial session is complete, such as *sprayd*.

Where To Get It

Many sites have copies of the latest version of TCP wrapper. An *archie* search revealed 96 sites with some available version. I retrieved my copy from *ftp.uu.net* in the */usenet/comp.sources.misc/volume36/log_tcp* directory. It's also available on *cert.org* and the author's archive on *ftp.win.tue.nl*. The *patchlevel.h* file should say it is version 5.1 after you apply *patch01*. The three *shar* files and the *patch01* only take up 154KB on your disk when uncompressed, and the final program is only about 25KB on the Sun.

Recommendation

Overall, TCP wrappers are an excellent tool for improving your network security. The logging features alone add a level of integrity that few UNIX vendors provide out of the box, and the ability to screen requests makes it harder to abuse services from outside your domain.

SAGE Views

Hierarchies in System Administration

by Wendy Nather
<wendy@il.us.swissbank.com>

How many of these system administration myths do you still hold near and dear to your heart?

- PCs aren't real computers; therefore no one who administers them can be a real system administrator.
- Real system administrators are male; females just aren't technical enough.
- Documentation writing isn't really technical work.
- If you can write device drivers, you should never have to change a toner cartridge again.
- Real system administrators don't have to deal with users directly; that's the Help Desk's job.
- Real system administrators are Hackers. Hackers are brilliant, eccentric hippies or nerds who don't have to have social skills because they're so smart.

The caste system is alive and well in system administration. The Untouchables at the bottom of the heap are the documentation writers, cable pullers, and customer support people who do all the work nobody else wants to do. At the top of the food chain are the old macho hackers who are pampered and indulged as thoroughly as any prima donna.

And if this admittedly sweeping generalization annoys you, stop to think about why it gets to you. Are you still seeing vestiges of this attitude in your own workplace? Does J. Random Hacker still oversleep and miss the group meetings without reprimand, or throw temper tantrums and slam-dunk his beeper into the toilet without reproach because he's the guru of the group? Do your coworkers somehow manage to let backup tapes go unchanged because they were busy writing poetry in *Perl*? Does all the documentation work end up piled on the desk of the female coworkers because "they're better at that kind of thing," while all the men shrug and say "I'm technical; don't expect me to be able to spell"?

I believe that in the realm of system administration, we cannot afford to drop the "dirty work" to the bottom of the list when defining a Real Sysadmin. A Real System Administrator should be able and willing to document when needed, pull cables with the best of them, explain something to a naive user with tact and diplomacy, manage time and projects responsibly, work cooperatively with others, and above all, treat other technical people as equals instead of perpetuating stereotypes.

For one thing, UNIX administration has opened up considerably in the past few years. Granted, it still requires a higher amount of technical expertise than most nontechnical management realizes, but it is also not the secret society it used to be. Management who didn't understand the inner workings of UNIX were heavily dependent on their gurus and indulged their idiosyncrasies since the technical pickings were very slim. But the workforce has grown larger, tools are making it easier and easier for lesser-experienced people to administer their own systems, and the industry is expecting more and better customer service from its system administrators. We can no longer afford to act as a privileged class, or make class distinctions among ourselves. We must concentrate on giving the best *generalized* service we can — user consulting, design, documentation, training, hands-on work with all other technical people, and an integrated system overview that only technical knowledge can provide. As Heinlein pointed out, specialization is for insects.

And as the old saying goes, if you want something done right, you need to do it yourself. Maybe if the most adept technical people wrote the documentation, we'd have good documentation we wouldn't have to complain about. If hot-shot hackers would devote time to installing and maintaining PC networks, the glamour and mystique would transfer itself to that job and would then be better compensated and appreciated. I have often seen higher-level problems noticed and solved by having a "guru" do the most mundane of jobs. There are very few tasks in system administration that cannot benefit from the expert's touch.

And finally, rather than getting yourself stuck forever doing the dirty work you're afraid to touch, you will increase your own value and marketability if you are willing and competent at every aspect of your job. I have seen several gurus passed over for opportunities they wanted because they were missing the requisite people skills, couldn't be trusted around customers, or had left a bad impression on management by their unprofessional behavior. If you work yourself into such a specialized, rarefied

position that you can't be replaced by someone else, then you also can't be promoted either. It benefits both you and your group to pass on what you know, rather than keep it to yourself to preserve your elite status.

Just as there are different types of intelligence, there are different types of technical expertise. Let's acknowledge and appreciate each sort because they all belong under the umbrella of system administration.

Why System Administrators Are So Short Tempered: Power Switches

by Elizabeth D. Zwicky
<zwicky@erg.sri.com>

I just spent 2.5 hours turning off computers. Admittedly, we have over 200 computers (and actually, I think our previous numbers were off, because I know we never counted any Amigas, and I know we turned off two). Furthermore, they are relatively evenly distributed over a floor- and-a-half of a very long building, which at the time was rather poorly lit. (The security guards turn off a lot of lights from the circuit breakers on the weekend.) On the other hand, there were three of us, which averages out to a little under 3 minutes per computer. As our waiting friends and relatives said as we struggled out of the building, weary but unbeaten, **"How can it *possibly* take *TWO AND A HALF HOURS* to *turn computers off*????!!!??"**

Well, to start with, no two models of anything have power switches in the same place. Within minutes, you develop switch-finding heuristics, which allow you to find most power switches rapidly while only looking moderately stupid. You grope for the power cord, and feel for a switch or switch-like object in its vicinity. Failing that, you push anything in that general area that feels button-like. (This has the happy side-effect of firmly seating all the fuses.) If the machine is still running, you glance at the front to see if there is a switch or button there. Still no luck? Try the right side to see if there is a big red switch. After this , you're down to feeling under things, cautiously feeling the rest of the back of machine, and then looking very carefully at the whole machine.

It's a good thing that the users aren't generally there to watch when we turn off machines. First, our heuristic mechanism generally takes longer on a machine with a big obvious power switch on the front, since you don't *look* there until you've tried the back. This looks stupid to bystanders, who can't understand why you don't just *look* at the machine. Looking at the front takes more effort than you'd expect:

- Nobody makes front-mounted power switches conspicuous on machines that sit in offices, because it would either annoy people or encourage them to push them at inopportune times. (This is usually a good thing; small children with a passion for buttons and switches do occur in offices.)
- Since most machines don't have front-mounted switches anyway, the sole purpose of such a survey would be to keep you from looking silly. It wouldn't work, either: after you turn off all the machines you can find, you stand in the middle of the office and listen intently to see if you can hear anything that's still running. This looks pretty silly any way you slice it, and even sillier when you start searching down the remaining noises.

Second, we don't exude that air of confidence and surety that users like to see. Users like to feel that you know what you're doing, particularly when it involves power, and discovering that you don't in fact know how to do something as basic as turning off the computer shakes their confidence. In point of fact, however, actually flipping the power switches is the only part of shutting down the facility that requires no expertise beyond a thorough acquaintance with the tricks designers use to hide the switch. The main trick is in knowing when you can flip the switch, and that we're very good at indeed.

Third, after 2.5 hours, you develop a sort of casual brutality about the entire enterprise. Can't find the power switch? Shove it around until you can see what you're doing. Still no switch? Turn off the power strip it's attached to. No power strip? Pull the cord off the wall and go on to the next. People think you ought to take the whole thing more seriously and slowly.

It's another one of these enterprises that causes you to develop firm and passionately-held opinions about things that other people regard as fundamentally trivial. For instance: power switches belong next to power cords. Power switches never, ever belong *under* things, even overhangs you can reach under easily. Power switches should be *switches*, with clearly marked on and off positions. Toggles are barely acceptable; rocker switches and flat plates that otherwise work like switches are unacceptable. Buttons are completely, totally unacceptable. Putting little diagrams next to them to show which position is on and which is off does not make them acceptable, particularly if on and off are distinguished by how far they stick up (How can you tell which picture matches the current state of the button?)

Slide switches are out. Knobs are beneath contempt, and a special circle in hell is reserved for people who design equipment which has a power switch but can actually only be turned off from software.

Furthermore, power switches should be the only switches next to the power cord. The Diag/Norm switch on Sun 3 machines is a cruel trick, particularly since it's much easier to find than the real power switch, and does nothing perceptible until the machine is rebooted, or not rebooted, as the case usually is. Any given piece of equipment should have only one power switch; putting one on the front that turns off most things, and then putting another on the back that turns off the power supply itself is not fair at all. Hiding power switches behind panels is not fair unless the panel is clearly marked as where all the switches are. Combining the two can only be ascribed to sadism. We had a tape drive at one point with a "Logic Power On" button on the front panel with all the other buttons, and the real power switch hidden behind the door you open to load the tape, all by itself. One of our hardware people disassembled the drive, figuring that it was broken, because the switch on the front panel was on but it wasn't getting any power. (Admittedly, this was the same drive that displayed error conditions in binary from right to left using the status lights, where blinking was one and solid on or off was zero, so it probably was designed by a sadist.)

For other reasons, I dislike lighted power switches — they're great when the little lights work, but what do you do when the light in the power switch burns out? To avoid this, our power strips use some neon-like bulb, which I assume doesn't burn out often. Instead, it flickers, thereby convincing the unwary that either the power or the power strip is intermittent. On the other hand, it is nice to be able to tell whether the object is turned on or not, without feeling for warmth or vibration. People look at you funny when you put your hand on pieces of equipment and then concentrate intently as if trying to commune with the machine. *You* know that you're trying to figure out whether it's vibrating because of its own fan, or whether it's transmitted from something else, or if it's not vibrating perhaps it's heating up, and if so is that because its fan is broken, and it's going to catch fire, or because it isn't supposed to have a fan and is just supposed to radiate heat gently, or maybe because the piece of equipment nearest it is blowing hot air on it and it's really turned off.

A few vendors have decided to remove the whole power switch concept by simply not providing the power switch, another unfortunate decision. The rumor says that in the initial design of the Kinetics FastPath (yes, when it really was Kinetics), they analyzed the mean time between failures of all the parts, and the ones with the shortest times were the power switch and the power lights, so they designed it out. They declined to recalculate the mean time between failures at this point, or they would have discovered that the power cord was now the most likely to fail, with a lower mean time between failures than the power switch had originally. This may all be fiction, but the fact was that the next hardware release had a power switch (and a power light), much to the relief of their customers.

Applications Software Development Symposium

by Rob Kolstad
<kolstad@bsdi.com>

On April 25-28, 1994, USENIX is offering a brand new kind of symposium: Applications Software Development. I'm very excited about this symposium, as applications development too often takes a back seat to the "exciting world of kernel development." If you're interested in submitting a paper (extended abstracts are due January 11, 1994), please contact Jim Duncan *<jim@math.psu.edu>* for more information. Hope to see you there!

Perl Practicum

The Swiss Army Chainsaw
by Hal Pomeranz
<pomeranz@aqm.com>

A Dab of Philosophy

I was expounding upon the glories of Perl to one of my colleagues the other day and he remarked that Perl seemed rather contrary to the UNIX design philosophy. Everybody has a different take on this issue, but combining simple tools to form more complex ones does appear to be a UNIX fundamental. Perl, on the other hand, intentionally provides a rich set a features which can, in turn, emulate a wide variety of different UNIX utilities. Of course Perl makes it easy to invoke various UNIX tools — it wouldn't be nearly as useful a language otherwise. Often, it will enhance the readability of your program to use different programs via `open()` or with backticks rather than trying to write the same function in Perl. There are real reasons, however, why this may be suboptimal.

First, there's the efficiency bogeyman. Certainly there is a great deal of overhead in setting up another process for execution. Of course, as the size of the data set you are processing grows, this overhead may become insignificant. As with all optimizations, you should experiment: try different solutions on real data sets and see which approach is most efficient.

A more telling argument in favor of avoiding vendor-provided tools is portability. If you have ever maintained software across multiple platforms, then you know how difficult it is to find the utility you need sometimes. Where does the *find* command live on all of your systems — */bin*, */usr/bin*, */usr/ucb*, some other evil hidden location? Does it accept the same set of options on all of your platforms? Multiply these issues by the number of different UNIX utilities that your Perl programs would like to use, and suddenly the cost of reimplementation appears much lower. If you maintain the same revision of Perl across all platforms, you have a consistent basis to work from. As further incentive, this installment will demonstrate some simple methods for emulating various UNIX utilities in Perl.

Cognates

Certain UNIX utilities translate directly to built-in Perl functions. Perl has built-in `chown()`, `chmod()`, `mkdir()`, and `rmdir()` functions. There's also `link()` and `symlink()` for creating hard and symbolic links respectively, as well as `unlink()` for removing files, and even a `rename()` function to partially emulate *mv*.

The `chown()`, `chmod()`, and `unlink()` functions will accept a list of files to operate on and return the number of successes. Sometimes, however, you want to know exactly which operations failed. In this case, use a loop over the individual elements of your list of files:

```
for (@files) {
 chown(0644, $_) ||
   warn "Can't change permissionson $_\n";
}
```

A similar strategy can be used for those file operations which do not operate on lists.

Note that if your operating system does not support one of the above function calls, you will encounter various failure modes (some more graceful than others). For example, if symbolic links aren't supported on your system, then the `symlink()` call will cause your program to die at runtime with a fatal error. Like any function you are unsure of, you should wrap the call in an `eval()` statement to trap possible errors:

```
eval "symlink($old, $new);";
warn "Symlink not supported\n" if ($@);
```

The `$@` variable is guaranteed to be the null string if the `eval()` succeeds, so this is a reliable test.

Sometimes Perl will simply invoke the appropriate operating system tool if a function is not provided as a library call: the `mkdir()` function is the classic example of this. In this case, it is probably more efficient to call the program once yourself with a list of directories, rather than spawning a process for each individual directory you wish to create.

Subtlety

Certain Perl functions are closely related to UNIX filters. For example, `split()` and `substr()` emulate cut very closely. Perl has a builtin `sort()` function that is much more powerful than the UNIX *sort* utility, but you have to define your own selection routines to do really tricky sorts. (See the first "Perl Practicum" in *;login:* Vol. 18, No. 5, September/October 1993, for more information on devious sorting). Sometimes, though, you have to change your thinking a bit to get Perl to do what you want.

For example, programmers often like to use *basename* and *dirname* to get the name the current program was called by (for error messages) and the directory it was called from. Perl stores the full invocation pathname of the program in the variable

$0 and *basename* and *dirname* can be emulated with appropriate substitutions:

```
($basename = $0) =~ s%.*/%%;
($dirname = $0) =~ s%/[^/]*$%%;
```

The first substitution takes advantage of Perl's greedy pattern matching algorithm to eat up everything up to the last '/' in the pathname and throw it away. If you're interested in both the directory and the file name, you can use the following one-liner:

```
($dirname, $basename) =
                    $0 =~ /(.*)\/(.*)/;
```

Again, we're making use of the greedy pattern match as well as the fact that pattern match returns a list of subexpressions in a list. The statement looks a little strange, but the precedence is correct.

Another common UNIX filter is *uniq*. Of course, you always have to sort your file before passing it to *uniq* because the tool will only recognize consecutive matching lines. Not so with Perl:

```
open(FILE, "< myfile") ||
  die "Can't open myfile\n";
while (<FILE>) {
  next if $seen{$_}++;
  ...do some processing here...
}
close(FILE);
```

Note that memory usage can get quite high if the file is large and doesn't have a great deal of repetition. On the positive side, the `%seen` array ends up having a count of the number of repetitions of each line, in case you care to emulate *uniq -c*. You can always run `sort()` on the unique lines in the file if you really wanted the lines to be sorted.

The `grep()` function in Perl can be used to emulate UNIX *grep:*

```
open(FILE, "< myfile") ||
  die "Can'topen myfile\n";
@lines = <FILE>;
close(FILE);
@found = grep(/$pattern/, @lines);
```

This, however, can be rather memory intensive for large files. Instead, simply operate sequentially:

```
open(FILE, "< myfile") ||
  die "Can't openmyfile\n";
while (<FILE>) {
  next unless (/$pattern/);
   ...process here...
}
close(FILE);
```

If you want a list of matching lines, rather than operating sequentially, just `push()` the matching lines into a list in the processing section. At least you save having to slurp the entire file into memory.

The Perl Library

As distributed with Perl pl36, the Perl library contains several packages which emulate useful UNIX utilities. Additional packages are available in the Perl archives on *coombs.anu.edu.au* (150.203.76.2). Be sure to check there before reinventing the wheel.

You use a package by first "requiring" it and then calling the functions it contains as you would any user-defined function. For example, the *ctime.pl* package provides a simple work-alike for the UNIX date command:

```
require "ctime.pl";
$date_str = &ctime(localtime);
```

Of course, you don't get the formatting string capabilities that some date commands provide, put you can always use `localtime()` and `printf()` to emulate this behavior.

Also in the easy to use category are the *getcwd.pl* and *fastgetcwd.pl* libraries to help you find where you are in the directory tree. The function defined in *fastgetcwd.pl* is more efficient because it uses `chdir()` to traverse a path up to the root, but you might not be able to get back where you started from once you `chdir()` out. For those of you who like the `$PWD` variable under the C shell, there's the *pwd.pl* library. Simply `&initpwd()` after requiring the library and then use the package-defined `&chdir()` function instead of Perl's built-in `chdir()`. The `&chdir()` function will continuously update the `$PWD` environment variable.

Far and away the most useful volume in the Perl library, though, is *find.pl*. The union of *find* options across all UNIX platforms throughout history is tremendous, but their intersection is often minimal. Writing a *find* command that works on every platform at your site can be a study in constraints. Actually, the *find.pl* library really exists to drive the *find2perl* program provided with the Perl distribution, but you can require *find.pl* directly in a program of your own devising. (The *find2perl* program will emit a complete Perl program which will exactly match the behavior of the *find* options fed to *find2perl* on the command line.)

The `&find()` function defined in the library accepts a list of file and directory names as arguments and will traverse all files and subdirecto-

ries just as the UNIX *find* command does. For each item in the traversal, `&find()` will call a user-defined subroutine `&wanted`. No arguments are passed to `&wanted()`; but `$dir` contains the pathname of the current directory, `$_` the name of the item currently being considered, and `$name` the full pathname `"$dir/$_"`. If `$_` is a directory, the `&wanted` function can set the variable `$prune` to *true* to stop the `&find()` function from descending into `$_`.

Beyond that, the processing done in `&wanted` is entirely up to the user. Judicious use of the `stat()` or `lstat()` function and the Perl file test operators can emulate most of the options supported by the UNIX *find* command. (Don't forget that the `$_` variable caches the result of the last `stat()` or `lstat()`, whether the function was called directly or via one of the file test operators). Of course, Perl is a much richer and more powerful language than the command-line syntax of *find*, so some extremely powerful effects can be obtained. For more guidance, run *find2perl* on some of your favorite *find* invocations and study the output carefully. (There are probably dozens of good candidates in */usr/spool/cron/crontabs/** alone.)

Plenty of other tools are available in the Perl library. In *look.pl*, there's a dictionary lookup that emulates the UNIX *look* command. There's even a *syslog* interface in *syslog.pl* in case you hate calling *logger* all the time. More libraries are being invented, posted to *comp.lang.perl*, and archived at *coombs* every day.

Enough Already

Hopefully by this point I've convinced you that Perl is more than capable of emulating most simple (and some complex, e.g., the *s2p* and the *a2p* translators that come with the Perl distribution) UNIX tools. Sometimes, though you really need to call some tool outside of Perl. For example, I have yet to find anything better than:

```
chop($hostname = '/bin/hostname');
```

So, next time we'll be talking about strategies for writing portable Perl scripts across a bewildering variety of UNIX implementations with subtly different pathnames and command behaviors. Tentative title to be, "The Thing I Love About Standards Is That There Are So Many."

True Science

Some Psychological and Ecological Aspects of Programming

by V. N. Podsvirov, Mius, Taganarog
Translated by Judith E. Grass
<jgrass@cnri.reston.va.us>

[Translator's Note: This paper, from a recent Soviet UNIX Users Group meeting, originally appeared in the Proceedings for the 2nd Soviet UNIX User's Group Conference, Sept. 1992, Vladimir, CSU. It is translated with permission of the SUUG and V. N. Podsvirov. This is an abstract of a talk given at the conference. Unfortunately the full text of the talk is unavailable.]

The appearance of programming, first as the art, then as the science and technology of programming, has given rise to a series of problems. Programmers themselves have become the first victims of programming. Today it is impossible not to be struck by the impression that there is a pervasive, creeping computer addiction caused, in my opinion, by immersing a person in the trance state that is induced by hours of staring at a computer screen.

The nature of existence itself can be seen as a consequence of the conditions a person creates around themselves. For us, the most interesting characteristic of a person's general being is the condition of his mind. If some instantaneous "snapshots" of a person's mental condition are made over some period of time, it is possible to arrive at a deeper model of mental function.

The discrete indicators of mental condition, depending on the classification method, can be conditionally related to a few kinds of states:

- Mental ease: comfortable, uncomfortable, intermediate
- Alertness: asleep, half-asleep, wide-awake, in a trance
- Affect on health: healthy, indifferent, harmful, very harmful, deadly

Unfortunately, classification systems of this kind cannot be mathematically precise, but they do yield a deeper understanding of the subject.

According to my observations, normal condition of a qualified programmer seem to be the following:

- Intermediate mental comfort (slightly tending towards uncomfortable)
- In a trance state (caused by concentrating on a complex program and the specific effect of the rapidly changing information on a display)
- Indifferent for health (slightly tending towards harmful)

Thus, a programmer "walks on the razor's edge" while he is working and constantly risks falling into conditions that are harmful to health.

Just try to communicate with a programmer when he is involved with some work that is both critically important and urgent. If the subject of your conversation is far enough away from the ongoing work, then it is most likely that you will observe a person with marked deviations from the norm.

It isn't bad enough that programmers are so absorbed by their problems that they enter into trance states for days or even weeks at a time, but they do not even realize it themselves. Of course this harms not only the programmers, but it also harms those around them because of the "contagiousness" of the trance state.

Unfortunately, one of the most wide-spread means to exit from a trance is the use of alcoholic drinks, etc., which are in turn capable of putting a person into trance states which differ from the previous one.

The absorption of a person into programs and computer games, which is intensified by color and sound effects leads to trance states which can be harmful or extremely harmful and even incompatible with living, in as much as the "brain" resources of a person are not unlimited and should be reinforced by special exercises.

Debate: To Certify or Not?

by J. R. Oldroyd & Rob Kolstad
<jr@opal.com> <kolstad@bsdi.com>

[Editor's Note: After a piece of email from J. R. Oldroyd, I challenged him to an email debate on certification: Should SAGE create a certification program? We exchanged email, extending the previous text with each round of debate. This article presents the various points as we debated them, in a point-counterpoint-point-counterpoint, ad nauseam format. As it turns out, no ex post facto editing was allowed — you see the debate here just as it happened in email. J. R. had both the opening and closing shots in this battle.]

Summary

J. R. Oldroyd favors creating a certification subgroup with SAGE that can help administrators know what they need to know and help companies know that administrators have the requisite skills for the various positions a company might offer.

Rob Kolstad disagrees, citing excessive costs, inability to train for standard environments, and potential liability problems.

Round 1: Oldroyd's Opening

When SAGE was formed, one of the working groups that was created was Sage-Certify, chartered with the goal of establishing a certification program for system administrators. Within the group, it quickly became apparent that even the concept of certification was not agreeable to all, and the group ended up spending a great deal of time discussing the merits of certification. In this debate, I wish to put forward the notion that the concept of a Chartered System Administrator is useful and beneficial to the industry and that SAGE would do well to develop and deploy such a program in the very near future.

Professional certification is the process of obtaining formal qualifications in a field. A qualified individual has the right to indicate the qualification by means of letters after his or her name (J.R. Oldroyd B.Sc. CSA, for example), and as such, is easily recognizable to all as one who has achieved a certain standing in the industry, in terms of level of knowledge and experience.

Until now, our industry has been fairly self-sufficient, in that an administrator may move from one organization to another, but in doing so would remain within the computing industry, or at least would be employed by a computer-literate employer. But, with the rapid expansion of the global Internet that the 1990s is bringing us, we see increasing call for administrative services by individuals from noncomputer businesses, such as publishers, newspapers, banks, insurers, airlines, travel organizations, printers, and even small businesses such as local restaurants and food shops. The list is endless. Being able to recognize a CSA will be as important to these folk as being able to recognize a CPA (a Chartered Professional Accountant).

Round 1: Kolstad's Opening

SAGE's founding statement did indeed establish the certification working group and it has certainly been controversial. I don't think the controversy surrounds the good intentions of certification or the potentially good things that surround certification, but rather the problems one encounters as one opens this particular incarnation of Pandora's box.

Certification has a noble history: medical doctors, lawyers, certified public accountants. All meet some standard of competency and performance, at least according to whatever kind of exam they take to achieve their certification. Interestingly enough, none of those professions takes less than four or five years of intensive college training, and the medical field requires another half dozen years after the bachelor's degree.

System administration, as a field, is unique. I can think of no other field that shares even a majority of its qualities (and I've asked the question of dozens of other technical and nontechnical types). The field is incredibly broad and deals with systems in time frames from milliseconds through months. It deals with components whose size is measured in bits through components whose aggregate is measured in gigabytes (or even terabytes). It deals with cold, calculating machines and warm, human people. It sometimes deals with life and death; it deals with the background color of someone's screen. It is a discipline which, when performed best, is virtually unobservable. It involves planning ahead at many levels

I'm concerned with certification for two main reasons:

1. I do not think it is possible to come up with a process that can make strong enough statements about a person's skills and suitability for a particular job
2. I do not think that SAGE (or anyone else, for that matter) wishes to get into the business of standing behind the certified members in cases of malpractice.

It is hard for me to envisage a process that somehow tests a person's knowledge of system administration across the broad spectrum that SAGE's members inhabit: PCs, MACs, workstations, minicomputers, servers, mainframes, and supercomputers. Imagine the problem of testing just for knowledge of how to back up those systems! Some sysadmins deal with people constantly; others deal only with their immediate supervisors. How does one certify in this environment? I occasionally think that the "merit badge" approach to certification may be useful ("Trained in PERL", "Backups: Level VII", "Ethics & Policies"), but I'm not sure how useful they would be in reality.

As for SAGE standing behind the process, the legal and potential financial implications of being sued are staggering. Imagine American Airlines going out of business because their Certified Backup Administrator screwed up and lost one year's reservations. They would be quick to assess blame and try to redress their loss. I would prefer to avoid this arena entirely.

What does the loss of certification cost the world at large? I guess it does not ease the hiring of system administrators (as if certified administrators would be interchangeable, generic parts—which I doubt). It doesn't require moving money back and forth from those getting certified to those doing the certification. It does not stratify the community into the Have's and Have Not's. I'm not sure the losses are particularly large.

I am sure, however, that the potential problems with certification far outweigh the potential gains.

Round 2: Oldroyd Draws His Guns

But consider the cost to American Airlines of not having a certification program around. How will they know that the individual they hire to plan, install and manage their network of distributed reservations servers is even at all qualified to do the job. With an uncertified individual, they run considerably higher risk that the job may be bungled. For a large company such as American Airlines, who are likely to hire a team of administrators, this will probably be detected early on, but for a smaller company, such as Toscanini's Ice Cream which may need only one administrator to run a small corporate net, the consequences could be more immediately serious.

There are ways of protecting against malpractice. In all fields that currently require certification, a certified individual is obliged to take out some form of malpractice insurance: an insurance which covers the individual against damage claims in the millions of dollars. The certifying agency, SAGE in this case, would provide this coverage to its members, and would maintain lists of members with and without current coverage. SAGE can explicitly protect itself against claims arising from damages caused by members who's coverage has lapsed. The parallel here is in the diving industry. Dive instructors and dive masters must have certification and insurance. The certifying agencies all maintain lists of members in good standing; a quick call to the relevant agency will verify someone's credentials

The wide-ranging nature of computer system and network administration activities does, indeed, raise the question of how one goes about creating a certification process that will produce a useful indication of competence. I believe very strongly, that certification is something that should indicate *experience*, in addition to ability.

There are two issues here:

- What skills should be mastered before one can obtain certification?
- How does one measure experience?

Because of the broad array of topics involved here — network configuration, interconnecting PCs, interconnecting MACs, adding users, installing software, training users, fixing printers, managing an Internet connection and fire wall, etc., etc., to name but a few — when one realizes that the administrative duties required by one organization may be quite different from those required by another organization, it becomes clear that a single level of broad certification will not be practical. Instead, a system whereby one obtains a base certification in a set of common subjects, and then adds specific "decals" in more detailed areas seems more appropriate.

The complete lack of standards in this area, has caused wide divergencies between even similar platforms. This means, adding a user on a SunOS box, is quite a different procedure to adding a user on an AIX box. Never mind a NetWare server. Whether or not you can add a user to all these boxes in one go, depends on which boxes you have, and even then, procedures may be different. Until standards are in place, and POSIX

and X/Open (amongst others) are working on this, the certification program will have to take account of this. Already, individual vendors are moving to deploy their own certification programs, clearly showing that they feel there is a need for certification.

I would anticipate an industry-wide certification program looking something like this

- Base Certification (in selected broad categories) to which could be added any of:
  - Decal in use of AIX admin tools
  - Decal in use of HP-UX admin tools
  - Decal in use of SunOS tools,
  - etc.
  - Decal in MAC internetworking
  - Decal in PC internetworking
  - Decal in Internet connection management
  - Decal in backup service provision
  - Decal in documentation tools (Frame, Interleaf, troff, PC/MAC tools and conversion between them)
  - Decal in supporting internal tools on multiple platforms; cross-compiling, and distributing.
  - etc.

Although at first, one may argue that the list of decals would be too large, in fact with a decent Base Certification, I don't think the list would be excessive. A candidate would not have to obtain all decals; just the ones that are relevant to his/her desired area of expertise. This is similar to a medical doctor specializing in ear/nose/throat or heart/lungs. If system administration is a broad field, I'd argue that the medical field is broader, and certification does work there.

As for measuring experience, this can best be done by devising a form of Master/Apprentice program. A candidate would have to demonstrate to one or more qualified Masters experience built up over time in the area in question. The Masters' recommendations, taken with the candidate's examination results, should give a valid measure of ability. The Master/Apprentice relationship need not imply that candidates actually work for Masters; in this day of highly-available communications, individuals could have Masters at other locations, possibly even quite remote; as long as the Masters are able to confirm the candidates' abilities and satisfy themselves of this, the system should work.

In the near future, when many noncomputer literate organizations will be in need of WAN and Global Network services, I believe that a recognizable qualified administrator will be essential. I believe that SAGE is perfectly positioned to offer such a program. I do agree, though, that there are a number of concerns against such a program; I feel, however, that a useful program can be designed that addresses all the concerns and works within the imposed constraints. If a dive instructor or medical doctor can obtain certification in a field where a mistake literally means you end up with a dead client on your hands, it is clearly possible to certify a computer administrator.

Round 2: Kolstad Shoots Back

I consider the notion that SAGE must bless American Airlines' administrators (and that neither AA nor TIC can determine their own requirements) to be awfully paternalistic and not particularly realistic. Surely there is in existence proof that AA can get along just fine with today's system administrators — they seem to be muddling along all right so far. As to the financial liability issue, I do not think that the solution to these kinds of problems is to back them with insurance. My insurance company (State Farm) will have nothing to do with software companies — they will not even insure my furniture against fire damage! Why? Because they believe that the amount of liability could be so excessive as to be unpayable. I agree with them on that aspect of this topic.

Regarding experience, I hear the concept of certification as providing some kind of promise that a certified administrator has, in fact, performed certain prescribed tasks. This sounds to me like an expensive undertaking (if the tasks/positions are to be certified in any real way — not just as "Lee Admin was employed for three years at an oil company").

As to the merit-badge approach to system administration certification, I believe that the diversity and lack of standardization will, in fact, require dozens of "decals" — or else "decal-modifiers" like "Backup on SunOS systems with local data storage and PCs running PC/NFS and 8mm tapes using commercial backup product XYZ on Ethernet and TokenRing." I'll argue that this isn't very much help.

The proposal for the Master/Apprentice system seems to me to have real potential problems of political cooperation rather than technical cooperation. I can too easily imagine, "We all like Martin,

so let's make sure he/she makes it through the certification process..."

I do not believe that SAGE is positioned to create a certification program. The concepts are incredibly wide-ranging for an organization like SAGE to attack. Furthermore, the expenses of creating and administering such a program will do nothing but grow, and will be exorbitant in the face of the meager benefits that will be provided.

Round 3: Oldroyd Gets Bigger Guns

Very large organizations do, indeed, have the resources to muddle along without certified staff. But nevertheless, I'm sure that even these organizations would prefer to use recognizable staff. And, the many smaller organizations out there would definitely prefer this.

It's not surprising that State Farm will not pick up this kind of liability. One insurance provider would be mad to accept this kind of coverage. As you say, it is too large for them. But, the large brokerage firms will handle this, by spreading the risk over several insurance providers. I have checked with Rollins, a large broker here in Boston, and they are only too ready to talk about SAGE's future requirements.

Regarding merit-badges like "Backup on SunOS systems with local data storage and PCs running PC/NFS and 8mm tapes using commercial backup product XYZ on Ethernet and TokenRing," well, yes, I agree. You'd soon end up with too numerous decals to be realistic, and what use, indeed, would they be? No, I believe the decal system should be more generic, with a decal for "Backup Service Provision," which would cover backups of DOS PCs to UNIX hosts in general, as well as general coverage of other backup environments. Sure, it won't cover everything; the candidate's own experience in that specific area would cover many gaps. The decal would indicate that a candidate has specific expertise in the areas of backup, and would be more suited to determining how to do backups on SunOS, with PC/NFS PCs to 8mm tapes, than would an administrator who handles, say, Internet connections.

As for the Master/Apprentice system, yes, it is possible for everyone to help Martin along by pushing her forward, but the examinations would also help filter that out. And, if Martin flunked because she hadn't really got the experience or knowledge, that would look bad for the Masters who sponsored her.

Let's take a look at what would be needed within SAGE to do this. I agree with your comment that SAGE is not currently set up to provide a certification program. But, they are the ideal group to do this.

A certification program will require a staff. Someone to manage the program, to develop the curriculum, and to work in conjunction with technical experts in the field to get industry approval of the program. Someone to implement an examination procedure and to coordinate and manage testing. Someone to coordinate the Master Administrators. Someone to provide certificates, registration, and verification. This is a major undertaking, and one which will take time, effort, and resources.

If SAGE is committed to providing this service, funds have to be provided. The funds needed will have to come, in addition to member contributions, from the industry. SAGE will have to look towards corporate sponsors who are willing to fund the creation of such a program. Despite your reservations, I believe that, with proper marketing of the benefits of a certification program, there will be enough sponsors out there willing to help make it happen.

Industry needs to be committed to this. After all, industry will have to make time available to employees to obtain the training and certifications. Industry will be one of the beneficiaries of the program. And, I see enough people out there asking me how to recognize a qualified individual to know that there are industrial sponsors out there.

Round 3: Kolstad's Final Rejoinder

You assert that big organizations desire certified staff; I'm not so sure this is true. I'm currently working with a house designer who is not a certified architect. His reputation exceeds that of most architects in this area and — important to me at one level — his charges are lower. I am unwilling to yield the notion that companies would rather have a certified than a noncertified administrator until we have agreed that certification can work and that certified administrators are somehow better. Until then, this is a bit of cart before the horse.

As to insurance, I'd sure like to quantify the costs before jumping in. The potential liability is so incredibly large (e.g., the loss of an entire multibillion dollar business), that the costs of insur-

ance could be dramatic. Why such increased costs should be borne — and who should bear them — are questions that need to be addressed before accepting assertions that the liability presents an acceptable risk.

I'm concerned about the your reply to the merit badge approach. The notion of Generic Skill coupled with (unverifiable) experience doesn't have much value to me.

The certification program you propose requires: management, staff, "industry technical experts," and many more. Of course, we haven't seen much success in the identical program that UNIX System Laboratories has founded to ensure that UNIX programmers are certified. It is unclear how they invented their examinations, but one thing is clear: acceptance is low and costs are high. I do not see an industry clamor for USL-certified programmers.

Financially, it sounds like certification can be as expensive as some manager wishes to make it: consultants, expert test creators, release time for people to become certified (presumably including required courses they have to take to fill in gaps), and marketing to convince people the program actually has value! Professionally, system administration (a profession which has a difficult time explaining what its members do, if my experience is any indicator) will join the very small number of other professions which require certification. It sure sounds like a lot of work to provide a benefit which, to me, is dubious. ("American Airlines can have confidence that their administrator can perform generic backups and claims to have done so in a networked Sun environment.")

In summary, I believe that the claimed benefits of certification revolve around reducing risk in hiring of administrators by big industry. I believe that the costs are claimed to be high — and that's only by the very first reckoning. (So few things get cheaper with time.) I do not believe that certification will dramatically ease the hiring process because so many other factors contribute to an employee's success: attitude, attendance, and ancillary skills, to name three.

I do not believe that the hiring of administrators and the performing of system administration tasks are processes so broken as to merit a fix of this magnitude. The benefits seem small, the costs seem high, and the initial high concept — "Let's just make sure all the administrators have good training and give them certificates to certify that in various generic areas" just doesn't bear much scrutiny before one realizes it will become a nightmare of bureaucracy, overhead, and complex technical wordsmithing. It will not be much benefit (to anyone) at all.

Half-Round 4: Oldroyd Summarizes and Exhorts

In saying that I believe certification has a place in our industry, I am thinking not only of now, but also the future. In recent years, I have come across more and more clients who view computer system and network administration as an unavoidable expense that they must endure. In the (near) future, with the increasing deployment of global internetworking and information services, we are likely to see an ever increasing demand for computer administration services. To some extent, I agree that the situation we have now is controllable enough that a major program such as SAGE certification is probably not necessary. But, I believe that we should act now, in order to be able to meet the demands that will be placed on our profession in the next two to five years, and this will mean providing a certification program.

The benefits of a certification program are real, and will become more important soon. The benefits accrue both for employers and for administrators. Employers will have an obvious indicator of individuals who are qualified. True, individuals without certification may be equally capable, but as the field expands, as it must over the next few years, there will be a market for certified individuals. From the individual's perspective, certification will be a benefit because it enables them to see what training they need and gives them a clear indicator of what is considered to be required and necessary skills. The individual whose certification level is broad ranging, indicating a wide skill set combined with practical experience, is a more marketable individual.

We have discussed the amount of effort that will be required to provide such a program. As I noted before, it is not trivial. I submit that there is a need for this effort, and that industry will be able to benefit from it. A logical next step would be for SAGE to seek out potential industry partners for this program, partners who will not only benefit from the effort, but who will be able to support the development of the program, both financially, and with provision of technical experts. Once a group has come together, the mandate for the program will have been demonstrated.

In the meantime, individual companies are going ahead with their own programs. Based on Novell's successful Certified NetWare Engineer program, both USL and Sun Microsystems are now offering their own UNIX and Solaris certification programs (respectively). These companies obvi-

ously recognize the need for certification and are supporting the effort. However, their programs only cover their own systems. As vendors, they would benefit by incorporating their programs into an industry-wide SAGE certification program: very few customers these days have only AT&T or only Sun equipment. You commented on the lack of acceptance of the AT&T and Sun programs; could this be due to their very vendor-specific nature in what is an entirely heterogeneous market?

SAGE does not have to commit initially to massive deployment of resources in order to create a certification program. A premarketing campaign, aimed at soliciting partners can be conducted first. This will not be too costly, and will have a finite lifetime. From there, it will be obvious whether or not there is sufficient basis to continue. Given, though, that other programs are emerging, if SAGE is to provide the industry-wide, vendor-neutral program that is needed, SAGE must act swiftly, before the market moves ahead and the number of diverging other programs is too great. The time to act is now.

[Editor's Note: If you'd like to debate a topic publicly with me or anyone else, please drop me <kolstad@bsdi.com> a line so I can share the publication constraints with you and your opposition. Or you can contact SAGE Working Group on Certification <sage-certify@usenix.org>.

;login: 50 and 100 Years Ago

by Barry Shein
<bzs@world.std.com>

December 1943
Future Meetings

There was a discussion regarding the possibility of changing over from the current two general conferences per year schedule to one per year, or possibly one per year and coalescing workshops as a program for the other general conference slot. Scooter Johnson raised the issue of lack of "underwoods" (typewriters) in academia as possibly affecting the quality of paper submissions. Boots Shein mentioned gas rationing as putting a damper on travel budgets. The issue was tabled for further study.

Directors Meeting Minutes

We are pleased to announce that Vannevar Bush has agreed to give the keynote address at our upcoming winter meeting in Hoboken, New Jersey. Mr. Bush is science advisor to President Roosevelt. He has chosen the topic "Notes in Swing Time" describing a bold initiative by our government to utilize broadcast radio to drive teletypes in an effort to make interoffice communications within Washington, DC more efficient.

Scooter Johnson
President's Report

December 1893

Although it is not USENIX's policy to address the morals of members, an issue regarding several incidents of debauchery at the past Summer Conference has been raised by our WCTU working group. As leaders of a world not yet come to existence, the question we have to ask ourselves is whether or not alcohol and technical recitals simply do not mix in our public activities, and what image we expect to project.

Letter to the Editor
Constance Wilson

A kit for designing one's own weaving patterns using Mr. Jacquard's techniques is now available from the *Sears and Roebuck Catalog*. We have arranged a discount for the USENIXmembership. Instead of the usual 85 cents our members can order this new product for 70 cents, plus the usual shipping and handling (5 cents).

Technology Update
Jarad Duncan

Perl 5.0 Overview

by Tom Christiansen
<tchrist@cs.colorado.edu>

The last major release of Larry Wall's freely redistributable Perl programming language was three years ago. Since then, Larry's released a few updates, but these have been almost entirely bug fixes and portability enhancements: no significant new functionality was introduced. During this time, Perl has spread from hard-core UNIX shops into regions never dreamt of by its author, from trading firms to chip designers to heavy industry.

Although its origins are firmly rooted in UNIX, Perl now runs more or less happily on operating systems as diverse as MS-DOS, Windows/NT, VMS, the Apple Macintosh, and many others. It has become the language of choice for harried systems administrators who haven't the time for slow, awkward, and unportable shell scripts, nor for inscrutable, tedious, and unportable C programs. Increasing numbers of companies are now shipping Perl and using it internally for their install programs, test suites, database interfaces, and end-user applications.

With its 5.0 release, Perl promises to address an even broader range of systems and applications programs than ever before. For certain applications, programmers will still labor long and painful hours writing in C so that they might squeeze that last bit of efficiency. Still, many of us find that a good interpreted language is plenty fast enough, especially on today's blazing hardware, which can only get faster quickly.

The new features in Perl 5.0 will not break existing code except in a few extremely rare cases where the writer was being, to quote one of C's fathers, unreasonably chummy with the compiler. This is because the internals have changed drastically, yielding a leaner, cleaner, faster, and more predictable program. The grammar is much smaller and simpler, and the language is more forgiving of mistakes and more informative when you do something questionable, instead of silently doing something you might not expect.

For example, you no longer have to remember which things must have, might have, or can't have parentheses, although the cautious programmer will continue to use them in all circumstances to make life easier for software maintainers. Even better, most of the surprise factors causing difficulty for new users, such as context sensitivity, now generate warnings (under the *-w* switch) when you're doing something that doesn't really make much sense, like multiplying two non-numeric strings together, using a list when you want a single element, and so forth. Running perl without *-w* would be like running classic C without using *lint*.

The single most important area which Perl now addresses is that of nested data structures and references. A given element of a list (linear array) or table (associative array) can itself be a reference to another list or table. This allows you to construct all the complex data types which you were longing for, such as binary (or n-way) trees, C-style structures and jump tables, or lists of lists of lists. For example, here's a list containing sublists:

```
[2, 4, [2, 9], 8,
    [8, 12, [2, 5, 0], 9], 8]
```

And here's a table containing other tables. (The => is just a glorified comma that visually distinguishes when you're constructing tables.)

```
{
    RED => { CRIMSON => 1, SCARLET =>2},
    BLUE => { AZURE => 1, INDIGO => 2 },
}
```

Furthermore, these can be mixed and matched; here's a table, indexed by someone's name, whose values are each lists of names:

```
{
    John => [ Mary, Pat, Blanch ]
    Paul => [ Sally, Jill, Jane ]
    Mark => [ Ann, Bob, Dawn ]
}
```

We now have references and referencing and dereferencing operators to go with them. References in Perl are type-safe and type-checked, unlike C's pointers. Furthermore, variables have reference counts on them to control when storage is released.

For postfix dereferencing, you have C's `->` operator, allowing you to write things like

```
$r -> [3]                       # list element
$r -> [3] -> [4] -> [17] = 3; # 3-dim array
$r -> {"John"}                  # table element
$r -> {"ru_utime"} -> {"tv_sec"} # nested table
```

The last demonstrates the most straightforward way to represent C-style structures. More elaborate methods are also possible. Multiple level constructs need not be declared — each level is created on the fly if needed, just as assigning

something to a scalar variable makes it spring into existence without any previously declaration.

To create a reference to a named variable, we use a backslash where C used an ampersand:

```
$sref = \$some_var;
$lref = \@some_list;
$tref = \%some_table;
$fref = \&some_func;
```

For prefix dereferencing for which C uses a `*`, you can use any of the four previous type specifiers: `$`, `@`, `%`, or `&`. You can use more than one of them, in fact, because you could have `refs` to `refs`, or functions returning `refs`.

```
$$sref = 3;             # assign 3 to $some_var
pop(@$lref);            # pop @some_list
@key_list = keys(%$tref);# keys %some_table
&$fref($parms);         # some_func()
```

As in the shells, you may always use braces to clarify:

```
pop( @{ $lref } );
&{ $jump_table{$name} }($keystroke);
${${$refref}} = 1;      # like $$$refref = 1
```

Finally, there's a `ref()` built-in function that returns the type of reference you've got, allowing you to write functions to print out nested data structures without knowing quite what's in them beforehand. Possible return values from the `ref()` are "" if it's not a reference, or `SCALAR`, `ARRAY`, `HASH`, `CODE`, or `REF`.

```
if (ref($r) eq "HASH") {
   dump_hashtable()
}
```

That's right: you don't need to use the ampersand on a function call anymore if you don't want to.

And yes, you may have user-defined data types as well! Using Perl's package system, you can write code using object-oriented programming strategies. Per-class and per-instance user-defined constructors and destructors are supported (reference counts are essential for knowing when to call the destructor), as are multiple inheritance and class-specific functions and data. For example:

```
$r->next_seq()
```

would call the `next_seq()` method of whatever class (package) to which `$r` belongs (`next_seq()` could get at `$r`, it's "this" pointer, in C++ parlance). If there isn't such a method, then a run-time search up the inheritance chain will ensue. If none is found, then an exception would be raised using Perl's exception handling mechanism. If one is found, then it's cached so that the search can be shorter the next time.

Here's another interesting possibility. These two are identical, making it slightly nicer to read and write certain constructs:

```
$new_ob = $old_ob->new();
$new_ob = new $old_ob;

$count = $ob->sizeof();
$count = sizeof $ob;
```

Nifty, eh? Those are user-defined methods, not built-ins. This isn't just gratuitous syntactic sugar: it falls out of the "indirect object" slot such as is found in output statements, sort routines, etc. It's really just like:

```
print $fh "string\n";
```

The other major set of functionality that will greatly aid programmers is in the area of scoping. Perl now supports both static and dynamic scoping of variables. Originally only dynamic scoping was supported, but so many C and Pascal programmers were unused to it that they found themselves making strange mistakes.

It used to be that variables always sprang into existence when you first used them whether you wanted them to or not. Now, you may optionally enforce variable "declarations" on a per-block basis. For such blocks, all variable references much be either a statically-scoped local or a fully-qualified global. Any stray variable references will be flagged at compile time. This makes it much easier to write safe code that doesn't accidentally alter or create a global variable.

There's quite a bit more we don't have time to go over in detail. Here's a partial list of what's already been completed that we haven't gone over already in this article:

- Nestability of quoted strings
- Improved exception mechanism
- Support for `BEGIN` and `END` subs on a per-package basis
- Various bug fixes

And here's a list of some of what's anticipated to be there, but not strictly guaranteed:

- Embeddability into C and C++: *cc prog.c -lperl*
- File handle objects: `$STDOUT->`*flush*(1)
- Separate man pages for all library functions and built-ins
- Very easy GUI Perl applications using high-

level X bindings

- Many more libraries, including *class* and *struct* libs
- More example code in the *eg/* directory
- A Perl profiler
- Debugger enhancements
- Access to POSIX 1003.1 functions
- Mnemonics for all the "funny" variables, (e.g., $ERRNO for $!)
- Easy extensibility using C functions
- Various Perl development tools

If you'd like to play with an alpha release of the Perl 5.0 release, you can retrieve it from *ftp. netlabs. com* [192.94.48.152] in *pub/outgoing/perl5.0/perl5a3. tar.Z* . It already contains all the functionality detailed in the long exposition above. It also has interesting files with more information: check out *Changes*, *Todo*, and *Wishlist*.

Automatic Information Server

The Association has a mail server that will answer requests received through electronic mail. Information is available about USENIX and SAGE membership, conferences, and publications.

Please send all requests to: <*info@usenix.org*>

To receive the *Primary Services Catalog*, the body of your mail message should contain only the line:

```
send catalog
```

This catalog outlines accessible information about the USENIX Association's services. You can get more detailed listings for subtopics in each service category by sending the line:

```
send service catalog
```

where *service* is one of the primary topics listed below. Thus:

```
send conferences catalog
```

will result in your receiving a listing of available announcements for upcoming USENIX Conferences and Symposia.

All files are in plain text format. The following subtopics exist currently:

Service: conferences
Description: The *conferences* catalog contains all the current announcements for USENIX Conferences and Symposia. This includes all available calls-for-papers and registration information (technical sessions, tutorials, fees, and accommodations) for upcoming events.

To get the *conferences* catalog:

```
send conferences catalog
```

Service: publications
Description: The *publications* catalog contains ordering information for our conference proceedings, back issues of our quarterly technical journal, *Computing Systems*, as well as publications, announcements, and calls-for-papers for special issues of *Computing Systems*.

To get the *publications* catalog:

```
send publications catalog
```

Service: membership
Description: The *membership* catalog contains information about the USENIX Association, the System Administrators' Guild (SAGE), and an application form.

To get the *membership* catalog:

```
send membership catalog
```

Service: papers
Description: The *papers* file contains information about submitting papers to upcoming USENIX conferences and symposia, including a sample abstract and advice on how to write a good paper.

To get the *papers* catalog:

```
send papers catalog
```

Questions, comments, suggestions: please contact <*info-admin@usenix.org*>.

PEM

Opinion submitted by Wm. Randolph Franklin
<franklin@iss.nus.sg>

The March/April *;login:* and earlier issues mentioned privacy enhanced email. With it, you might send your email, encrypted, to a router, which would strip the header, decrypt the real addressee, or the next router, and forward the email. Assuming that the router did not record its actions, this would make it harder for an eavesdropper to trace email.

Clearly, such a capability is as desirable as the capability to send papermail without a meaningful return address. Legitimate uses, for example, might include sending credit card numbers. However, there are problems:

1. If only a small fraction of email uses the router, then an eavesdropper will target mail to and from it, as being the most interesting mail on the net. Also, unless the router delays mail for a random length of time, and changes its size, traffic analysis will be possible. Knowing who sent a message, but not its contents, may be sufficient. Consider if I'm expecting a review of a proposal. Knowing who the project director is communicating with could be useful to me. If I know a message's sender and length, I might be able to match it to one of the anonymous reviews that the director forwards to me.
2. The law considers certain legal actions as so illogical that only a criminal would do them. Renting a safe-deposit box under a false name, or storing cash in a safe-deposit box, are examples. If you do either, you are guilty to the IRS and Tax Court until you prove yourself innocent. If only a few people used privacy-enhanced mail, then big brother might argue that only someone with something to hide would use PEM.

The best solution to these possible problems would be for everyone to use PEM as widely as possible if it becomes available.

Pragmatica: The Alias Builder

by Lee Damon
<nomad@network.ucsd.edu>

This article describes a simple solution to the problem of how to decide where to send someone's mail when they have more than one available mailbox. It describes a set of utilities collectively called *Alias Builder* or *The Address Imploder.*

These tools are also useful for sites with a well controlled name space, as they allow users to control where email goes, freeing site administrative staff from the drudgery of this task.

Problem Statement

I recently began working in an environment where users might like to read their mail on a UNIX box, a Mac, a PC, or even an IBM mainframe. Each of these has its own operating system, login-id and name for mail delivery. Users might send mail from any or all of the these hosts at any time.

Additionally, no record of where people wanted to read their mail had been kept. I needed a way for anyone to send mail from anywhere in the company and know that it might be delivered to the right place. My major goal was to provide each user with a definitive mailbox location.

I decided to build a list of all the users and where they wanted their mail delivered.

The logical steps involved in the decision to write Alias Builder are fairly easy. I'm sure others have done the same. I went from contacting users and asking them about their mail preferences, to posting notes and having users send mail to me, to having them send mail to a program that would add them to a database.

The first draft of the program did only one thing. It received mail from a user and made the place that mail came from the user's mail delivery point. No commands, no options. It maintained a file called *aliases.db* which was written in the standard UNIX *aliases*(5) format.

I then created an *aliases.header* file with comments to be attached to *aliases.db*, and an *aliases.bulk* file to contain mailing lists. The *sh*(1) script *build_aliases*(8) puts the three together. All of these files are *rdisted* from a central system when they are changed.

Then came the hard part. How to make mail to the IBM Mainframe, Macs and PCs work with this system. We obviously needed a way to alias accounts from one system to another. The Macs and PCs could be grouped as single hosts, as they already have a multihost mail system — all that is necessary is to determine the host type to which the mail is to be delivered.

Architecture of the Solution

The main idea of Alias Builder was to make it easy for people to register their aliases. I wanted anyone in the company to be able to use it without having to call me.

However, I couldn't think of a way to have multiple aliases/systems per user registered in a sane manner, and I wanted to be able to do two other things — send help, and send a list of registered aliases. I bit the bullet and made the system driven by commands contained in mail messages.

In order to keep things simple, I decided to ignore the `Subject:` line and put the commands in the body of the message. By keeping out of the header, I was able to avoid problems with RFC-822.

I was able to get the functionality I wanted with only three commands and a default behavior of "send help." I made the command words case insensitive, and made them unique by requiring they be the first thing on a line and be terminated with a colon (:). The entire system was written with an eye towards not doing the wrong thing by accident. Only by issuing the right command in the right format will something be actually changed.

The commands are `Register-Me:`, `Alias-Me-To:` and `Tell-Me:`. You can send the `Tell-Me:` command with either of the other two and it will report what your new entries look like.

The actual steps that Alias Builder goes through are described in the "Implementation" section.

Operation

When mail is sent to "aliases" with no commands, or with unparseable commands, the system sends back a help message explaining how to use it. (See Figure 1.)

`Tell-Me:` causes the system to send back notes saying what users' primary aliases are, what their secondary aliases are, and what mailing lists they are on according to *aliases.bulk*. (See Figure 2.)

If a user sends a message with the line `Register-Me:` it will register the login-id and mailbox address in the `From:` line as the final mail destination. This is sent from the host (or a host of the same type) where the user wants to read email. (See Figure 3.)

If `Alias-Me-To: <alias>` is sent, the Builder will register the name on the `From:` line as being aliased to the *<alias>*. It will check to be sure the target *<alias>* is registered first.

The user first submits a `Register-Me:`, causing the system to register that login-id and address as that user's official mailbox. The user then sends `Alias-Me-To:` lines from the other host types, thus having all of his or her mail wind up in the mailbox on the right host (or host type, in the case of Macs or PCs), regardless of where it was sent or which of the user's login-ids it was sent to. (See Figure 4.)

The use of `Register-Me:` always overrides a previous `Alias-Me-To:`. If a user has a final delivery name of, e.g., `foo: foo@bar.bargle.com` with an alias `bar: foo`, and then sends a `Register-Me:` from the account `bar` on the host `glib`, the alias `bar: foo` will be replaced with `bar: bar@glib.bargle.com`. After that, the user can send `Alias-Me-To: bar` from the account `foo` on any host and will wind up with `foo: bar`.

The reverse isn't quite as true, however. Because `Alias-Me-To:` ensures that a user's mail is directed to a valid delivery name (including a host to deliver on) the user can't over-ride a previous `Register-Me:` until he or she has a new place to point the alias. (See Figure 5.)

This lets people change where their *One True Mailbox* is at any time.

The *aliases.db* file created by Alias Builder is fine for systems that use *sendmail*, or recognize the *aliases*(5) format. However, Microsoft Mail (commonly used on Macs and PCs) and Office Vision (on the IBM Mainframe) don't understand this format.

Unfortunately, I left the company before we found a way to convert the aliases file into something usable by these other systems. Right now, anyone can route mail via UNIX and know it will get to the right person, but the same can't be said of the other host types.

Implementation

Mail is sent to *aliases* which is forwarded to a specific host where it is piped into the *sh*(1) script *build_aliasesdb*(8).

The *build_aliasesdb*(8) first uses *grep* to find the lines with the command key words, as well as the `From:` line. It saves each command in its own variable name, and then proceeds to break up the `From:` line into a tuple consisting of the user name (login-id) and the entire address. The comments field (usually the full name, but sometimes a witticism) is ignored since all Alias Builder is interested in is the email address of the sender.

Next the system does a sanity check to be sure that the user didn't try to register the current address both as an *address* and an *alias* in the same piece of email. This would be selfcontradictory, and could cause some interesting mail loops in systems that didn't pay attention.

The program then looks at each of the command variables it had set earlier. For each of the possible three commands it enters a subsection and process it:

`Register-Me:` is the easiest. It calls the back-end program *ba*(8)with the tuple it found earlier (login-id, email address).

`Tell-Me:` does a bit more work. It first checks to see if the mail came from the user's primary mail host, or an alias. Once it knows the real email address for the user in question, it *greps* for that in the *aliases.db* file. It then looks for aliases in the file that point to that entry. Next it looks in *aliases.bulk* to see what mailing lists the user is a subscriber to. This last part is imperfect, as it won't follow alias trees. It only looks for the `alias:` name entry (See Figure 2).

`Alias-Me-To:` does the real magic. It is the only command that requires a user-supplied parameter. The command says to "take the email address from which you received this message and alias it to the following address." For example `Alias-Me-To: nomad` sent from *Lee_Damon@macmail* would write an entry `lee_damon: nomad` (data is stored in lower case) in the *aliases.db file*. Before storing it, the program makes sure the target address is registered. It it isn't, it sends email back to the user saying that the command failed and why.

One benefit of doing it this way is that a user can't register someone else's name and have the other person's mail go awry. You can send your mail to someone else, but not the other way around. Partial security, admittedly, but better than none at all.

Both `Register-Me:` and `Alias-Me-To:` call the *sh*(1) script *ba*(8) to do the actual writing. *Ba*(8) first checks for a lock on *aliases.db* and if it is busy spins until the file is available. This enables multiple users to mail to the Builder with out stomping on each other.

Ba(8) locks the file by creating a directory in */tmp*. It than stuffs a pid file into the directory, so a stale-lock check can be done. The lock is a 'while' call on the create of the dir — if it fails, the process sleeps for five seconds and then tries the create again. (Thanks to Greg Rose <*ggr@acci.com.au*> for the locking code.)

The lock also allows the administrator to edit the *aliases.db* file, should that ever be necessary. By creating the lock directory and stuffing the pid of your editing session into it, you keep *ba*(8) from trying to write while you are working on it.

Once *ba* has the lock, it does a *grep -v* of the file to delete the old entry if one exists, then adds the new entry. Lastly it runs the entire file through *sort*. It isn't often that you look at */etc/aliases*, but when you do, the sorted order makes it easier to find things quickly.

Once *ba*(8) has done its work, it exits. Any error code is passed back to *build_aliasesdb*(8). If the build was successful, *build_aliasesdb*(8) sends mail back to the user telling the user what was done.

Alias Loops

The problem with most aliasing schemes is the lag between one host getting updated and the others finding out. It is quite possible to start a mail loop between hosts when one host doesn't get the update right away.

Alias Builder tries to work around that problem by not updating the hosts directly. Each host receives the three files (*aliases.db*, *aliases.bulk* and *aliases.header*) via *rdist*. On each host, *cron* fires off a *make -f Makefile.namespace* every hour which calls *build_aliases*(8).

This way, all of the hosts get the new aliases at the same time, unless they are down when the *rdist* occurs or when the *make* should have been done. In such a case the update will be done within an hour and a half of the host returning to service.

Other Alias Control Systems

Perry Metzger and I built a system called *chmbox* when we were working for IBM Watson

Research. It solved the problems we had there, but was unsuited for the tasks at Gulfstream because it couldn't handle the idea that someone could have more than one alias to which mail might be sent. Users issued the command `chmbox` and gave the name of the system they wanted to read mail on.The *chmbox* program would then talk with the central *chmboxd* and have it update its database. *Chmboxd* would try to tell the target machine that the user wanted to read mail there, avoiding the mail loop. All other systems were notified via *rdist*.

Probably the best-known alias handling system is Majordomo. It solves the inverse problem, that of taking one name and bursting it out to many addresses. I call Alias Builder an imploder, because it takes many names and implodes them to one delivery point. Some day I hope to merge the files created by MajorDomo with Alias Builder to create a complete alias control scheme.

After I wrote Alias Builder, I discovered two publicly available solutions. Mailreg comes from UCSD and the other from a university in Queensland. I'm told they are available via *ftp*.

Several other people have mentioned that they too have programs to control their alias systems. I've encouraged them to write a paper like this one, and submit it. If you don't tell people about it, no one else will know that it is real and useful.

Availability and Portability

The Alias Builder package (a *tar* file containing this document, *ba*(8), *build_aliases*(8), *build_aliasesdb*(8), and the man pages for all commands) is available via anonymous *ftp* on *ucsd.edu* as *pub/alias_builder.tar.Z*.

In order for the *aliases.db* file to be written, *build_aliasesdb*(8), *ba*(8), and *aliases.db* should be in the same group as *sendmail*(8)'s effective GID, and *aliases.db* needs to be group writable. This is because *sendmail*(8) will be calling *build_aliasesdb*(8) with that group, and we don't want to make *aliases.db* world writable. If *sendmail* runs with *mail* as on many SysV boxes, it is easy. Otherwise, you have to find out what group *sendmail* runs under (typically *daemon* or *bin*) and make sure that group can write *aliases.db*.

Before use at any other site, the help messages that *build_aliasesdb*(8) sends need to be localized. They all reference the hostname that the system runs on. There is also one *sed*(1) line that has `gulfaero.com` embedded in it. Other than that, *build_aliasesdb*(8) should run just fine anywhere.

You may wish to reword the help messages. I tend to take a "friendly" tone in mine, and some sites will probably want something more formal.

Where Do I Send Comments?

Since I've left Gulfstream, and my new site already has an alias control system, I don't think I'll be doing more development work on the Alias Builder any time soon. However, you are welcome to send me email at <*nomad@qualcomm.com*> or<*nomad@network.ucsd.edu*> if you wish to discuss it.

Figure 1:

The help message, in reply to mail sent by *nomad@vanyel.eng.gulfaero.com:*

```
Date: Wed, 30 Jun 93 10:40:12 -0400
From: daemon@gulfaero.com (The devil himself)
Message-Id: <9306301440.AA19831@djini.eng.gulfaero.com>
To: nomad@vanyel.eng.gulfaero.com
Subject: Alias registration help information

Dear nomad,

You have sent me email asking for help. You may have sent me an
ambiguous command, or you may have sent one note with both a
Register-Me: and an Alias-Me-To: line. Anyway, here's the help message
for the Alias Builder, accessible to you by mailing "aliases@djini".

I work by figuring out where you sent me mail from, and modifying my
database to reflect that email. In this case you mailed me from
      nomad@vanyel.eng.gulfaero.com
```

```
so I would make changes effecting the nick "nomad" and
the address "nomad@vanyel.eng.gulfaero.com".

You issue commands to me by sending them as email to "aliases@djini".
Don't worry about putting a Subject on the message, just put one of my
commands in the body of the message and send it to me. You probably
shouldn't sign the note, as your signature might confuse me. (I'm only
a daemon, after all.)

The three commands I understand are:

      Alias-Me-To: <nick>
      Register-Me:
      Tell-Me:

      The Mailer Daemon
```

Figure 2:

A sample reply to the `Tell-Me:` command sent by <*nomad@vanyel*>:

```
Date: Tue, 29 Jun 93 16:44:38 -0400
From: daemon@gulfaero.com (The devil himself)
Message-Id: <9306292044.AA04336@djini.eng.gulfaero.com>
To: nomad@vanyel.eng.gulfaero.com
Subject: alias query results

Dear nomad,

You asked me to tell you what aliases were established for you.

Your primary alias: (This says where your final mail delivery occurs.)
--------------------
nomad: nomad@vanyel.eng.gulfaero.com

Cross-Platform mail aliases: (these point at your primary address)
----------------------------
knomad
lee_damon

Gulfstream-wide Mail Aliases: (mailing lists to which you belong)
-----------------------------
postmaster
reqadmin-generic
unix-supt

Please note that any commands you sent with this Tell-Me: request
will be processed separately. You should receive results from them soon.

      Thank You
      The Mailer Daemon
```

Figure 3:

A sample reply to the `Register-Me:` command, sent by <*nomad@vanyel*>:

```
Date: Wed, 30 Jun 93 10:40:25 -0400
From: daemon@gulfaero.com (The devil himself)
Message-Id: <9306301440.AA19893@djini.eng.gulfaero.com>
To: nomad@vanyel.eng.gulfaero.com
Subject: alias change

Dear nomad,

I have changed your system-wide alias in the master database. Please
allow up to 5 hours for the change to propagate. Please also make sure
you don't have a .forward file in your UNIX home directory, or if you
do that it doesn't send your mail to another host, as that can cause
serious mail loops
[...]

      thank you
      The Mailer Daemon
```

Figure 4:

A sample reply to the `Alias-Me-To:` command, sent by *<lee_damon@macmail>*. The command given was `Alias-Me-To: nomad.`

```
Date: Wed, 30 Jun 93 10:45:26 -0400
From: daemon@gulfaero.com (The devil himself)
Message-Id: <9306301445.AA20070@djini.eng.gulfaero.com>
To: nomad@vanyel.eng.gulfaero.com Subject: redirecting alias change

Dear lee_damon,
I have registered a forwarding alias for you. Mail sent to
lee_damon will be redirected to go to nomad.

nomad is in turn registered for final delivery
to nomad@vanyel.eng.gulfaero.com.

If this is not what you wanted, you can send another note to
aliases@djini with the corrected Alias-Me-To: line

[...]

        Thank you
        The Mailer Daemon
```

Figure 5:

A few examples of *Alias Builder* Transactions.

user: Jane Doe
UNIX Account: *jdoe@blarq.oil.com*
Mac Account: *Jane_Doe@macmail.oil.com*

Example 1:

First Jane thinks she wants to read her mail on the Macs. She logs into a Mac and sends mail to *aliases* with the single line

```
Register-Me:
```

causing the *Alias Builder* to put the following line into *aliases.db*:

```
Jane_doe: jane_doe@macmail.oil.com
```

It is stored in all lower case, just to make comparisons easier.

Jane then gets back a piece of email saying that she is registered.

Then Jane logs into any UNIX host as *jdoe*, and sends mail with the single line

```
Alias-Me-To: jane_doe
```

which causes *Alias Builder* to put the following into *aliases.db*

```
doe: jane_doe
```

and then send her another note saying that the alias was registered.

When all of this is done, Jane's entries in *aliases.db* look like this

```
Jane_doe: jane_doe@macmail.oil.com
jdoe: jane_doe
```

Example 2:

Now Jane realizes that she really wanted to read her mail using MH on her UNIX workstation. She decides to change her account. First she logs back into the Mac, and sends:

```
Alias-Me-To: jdoe
```

Alias Builder objects, saying that "jdoe isn't a valid destination" because jdoe doesn't list a host for the mail to go to, just an account. Jane logs into jdoe on her workstation, *blarq.oil.com*, and sends:

```
Register-Me:
```

which causes the *aliases.db* file to look like this:

```
jane_doe: jane_doe@macmail.oil.com
doe: jdoe@blarq.oil.com
```

Lastly, Jane sends mail from her Mac saying:

```
Alias-Me-To: jdoe
```

When all of this is done, Jane's entries in *aliases.db* look like this:

```
Jane_doe: jdoe
jdoe: jdoe@blarq.oil.com
```

An Update on UNIX-related Standards Activities

by Nicholas M. Stoughton
USENIX Standards Report Editor
<nick@usenix.org>

IEEE Standards Board

Mary Lynne Nielsen <m.nielsen@ieee.org> reports on the June, 1993 meeting in Montreal, Canada:

This time around, the IEEE Standards Board meeting was held in Montreal, Canada. Not only is it the home town of the current vice-president of the Standards Board, Wally Read, but also having a meeting in this location continues a tradition of trying to hold IEEE Standards Board meetings outside of the United States and the East Coast in particular.

Electronic PARs

NesCom, the New Standards Committee and the group in charge of reviewing Project Authorization Requests (PARs) for the Board, has slowly begun to nudge itself into the electronic arena by working on the development of an electronic PAR form. There has been a call for this for some years now. As such, Jim Isaak and Steve Diamond, two members of NesCom who represent the Computer Society, have been working with IEEE staff to develop this. A first draft was submitted for review and comment at this meeting. With any luck, standards developers will be able to submit PARs electronically sometime in 1994; some sticky issues, such as having real signatures on the form, are the reasons for the time delay.

Other NesCom news involved the approval of four new PARs to expand and rearrange the 1238 work (which also involved the earlier withdrawal of one 1238 PAR). Another PAR, 1003.11, was also withdrawn. This action was the result from an initial ballot on this standards profile on transaction processing, which demonstrated that the work was premature. As such, PASC asked that the PAR be withdrawn, and NesCom approved that request at this meeting. Details of all of this are listed below.

IntCom JTC1 Guide

As mentioned in earlier reports, the IEEE Standards Board International Committee (IntCom) recently approved a document to help IEEE working groups coordinate their standards development with ISO/IEC JTC1 (the international committee in charge of developing information technology standards) if working groups wanted to move their standards into the international arena. This guide was disseminated to a wide mailing list, and some concerns were raised about how the role of national bodies was represented. Because of this, IntCom agreed to prepare a revision to address this issue, and a revised guide will be prepared for the fall.

ProCom Actions

The Process Committee, ProCom, approved some new language to identify more clearly the role of organizational representatives, and this will be included in the ballot of new material for the *IEEE Standards Board Bylaws* and *IEEE Standards Operations Manual* for next year.

ProCom also had a long discussion about the merits of company membership and whether the IEEE should be examining this as a possibility. As most of you know, the IEEE is an individual membership organization. However, some groups have approached the IEEE about using its system of standards development, only to be turned down because they wanted to have company members in addition to individual members. (Remember, this is distinct from organizational representatives, which cannot be individual companies.) This, in turn, has led to some discussion over whether some working groups should be allowed to have company membership in their balloting, and if that should be restricted to all company membership, partial company membership, or some other mixture.

Needless to say, ProCom felt it had to gain a great deal of input before it could make a decision on this matter. As such, they are soliciting opinions and white papers on this subject. If you are interested, you can relay your concerns to the IEEE Standards Department, which will pass them on to ProCom. Some of the material presented to ProCom is also available (such as a summary of an earlier IEEE forum on this subject).

New Horizons

Finally, the IEEE heard reports on two areas of potential growth. One has been in the works for a couple of years now, the IEEE SPAsystem— the electronic bulletin board, delivery, and standards development system. At the June IEEE Board of Directors meeting (this is the board that runs *everything* in the IEEE, and that oversees the IEEE Standards Board, among other things), a loan

was granted to the IEEE Standards Department to support the implementation of the SPAsystem. This will go a long way towards helping get this large and ambitious project off the ground.

In addition, the IEEE Standards Board approved work on developing pilot programs in the area of conformity assessment. Unlike the SPAsystem, this is a brand-new area of work for the IEEE, which will hopefully increase the visibility and build the value of IEEE standards while also allowing easier entry of products that conform to these standards into international markets. Work is currently underway to pinpoint the pilot projects and to develop these programs further.

One other minor but interesting point: this was the first meeting in over a year in which PASC did not have at least one project approved at RevCom!

New PARs

P1351: Draft Standard for Information Technology–OSE Application Program Interfaces (APIs)–ACSE and Presentation Layer Application Program Interface [Language Independent]

P1352: Draft Standard for Information Technology–Test Methods for OSE Application Program Interfaces (APIs)–ACSE and Presentation Layer Application Program Interface [Language Independent]

P1353: Draft Standard for Information Technology–OSE Application Program Interfaces (APIs) C Language Interfaces-Binding for ACSE and Presentation Layer Application Program Interface

P1354: Draft Standard for Information Technology–Test Methods for OSE Application Program Interfaces (APIs) C Language Interfaces–Binding for ACSE and Presentation Layer Application Program Interface

Withdrawn PAR

P1003.11: POSIX based Transaction Processing Applications Environment Profile

Report on POSIX.0: Guide to OSE

Kevin Lewis <klewis@gucci.enet.dec.com> reports on the October 18 - 22, 1993 meeting in Bethesda, MD:

This was a week of shifting expectations. We were hoping to start resolution of the recirculation ballots, but, since the deadline was set after the ending of the meeting (and, by the way, was subsequently extended to November 8), the group shifted its attention to two other questions: "Where do we go from here, if anywhere?" and "How can we be sure that this document moves smoothly through the international standards community?"

We spent a fair amount of time talking about where we might go from here. Member of two groups presented their ideas in the form of sample PARs. One discussed ODP and the other focused on profiling from the user perspective. If you want my opinion (and you must if you're reading this), ODP seems to be the way that many think we should go. That doesn't mean that other work would not also be pursued. The problem is that old classic conflict of too big a plate for too few people, particularly given the state of the industry and the reduced number of participants.

One member was quite vociferous about having the group pursue ODP. The group's consensus was that it is not ready to pursue any other work as a group until the current effort is close to completion. But that doesn't mean that someone can't go off and champion such an effort alone, or with the help of others, and my belief is that someone may do this. I must also admit that the current reduction in participants along with the possibility of POSIX, or rather the Portable Applications Standards Committee (PASC), being restructured does, in my view, have a psychological effect on how new work might be accomplished. As you can see, there are many unanswered questions.

As for the question about getting the guide through the international standards community, the first hiccup occurred when SC22 omitted the POSIX.0 guide from its list of documents to be forwarded for Committee Document (CD) registration and letter ballot. Most of what I heard about this was in the halls and over lunch. It appears that SC22 did not like what it perceived to be a change in scope in Draft 16, the recirculation draft. The WG15 Technical Advisory Group (TAG) agreed to back up a step and forward draft 15 for SC22 CD registration and letter ballot. Yes, we are regressing ...) but, the plan, or should I say hope, is that all concerns raised by SC22 will be addressed during resolution of the recirculation ballots. I think they will be addressed ultimately.

The plan is to come into the January meeting at Irvine prepared to commence the recirculation resolution. I will be sending out ballots to the section leaders as I receive them so they can get started as soon as possible.

Now, for that $64 question: when do I think POSIX.0 will be done? I have been burned more than once trying to answer that. But I will try again: look for a completion, i.e. submittal to the IEEE standards board for approval, sometime during late summer or early fall of 1994.

The Bookworm

by Peter H. Salus
<peter@usenix.org>

Anticipation of Thanksgiving and Christmas is upon me as I write. So I'm going to be Mr. Niceguy and only talk in this column about volumes I really like.

Sendmail for Christmas

First of all, there's *Sendmail*. Those of us who have shuddered at a *.cf* file or wondered about the arcane mysteries of delivery agents or headers can now rest easy. (Or as easy as you can rest after reading more than 800 pages.) In the decade and a half since Eric Allman created *sendmail* (a.k.a. *delivermail*), I am certain that all of us use his program (or a descendant) daily. Eric, together with Neil Rickert and Bryan Costales have turned out a terrific book, complete with a genuinely useful tutorial. Furthermore, it's one that I was able to read. I'm sure tired of semi-literate (semi-illiterate?) technical books. I'm beginning to think that at most publishers the editors acquire, but don't actually edit.

Over the Internet

I would guess that much of the mail one produces goes across some segment of the Internet, in its broadest sense, though a reasonable amount must be sent between machines on the same Ethernet or LAN. And if what I read in the press and hear on television is even remotely close to reality, we have an ever-waxing set of computer-illiterate users trying to send mail, use resources, and overburden administrators. There's thus a hot market out there for folks like Brendan Kehoe and Ed Krol. The latest entry in the race is Dan Dern's *Internet Guide for New Users*. This is not a neat, slim volume like Kehoe's. It is large-format and over an inch thick. But Dern has managed to put a lot into this book, and I think the result is worthwhile. In addition to the expected material on getting connected, mail, newsgroups, bulletin boards, etc., Dern has a nice section on nettiquet, a good chapter on archie, Gopher, HYTELNET, WAIS, and their pals, and another on commercial services like GOPHER, HYTELNET, LEXIS/NEXIS, Dow Jones, and Mead Data. It may be the book you want to keep on hand for those too-frequent times when someone wanders in to ask how to *ftp* something.

Gigabits at a time

I went to hear a talk by Craig Partridge about a month ago in which he stated that gigabit networking wasn't tough. So, when I got home, I settle down over a period of about a week and read *Gigabit Networking*. I'm convinced that he's right! And I'm also sure that none of Al Gore's advisers have read the book. They should. In fact, I'm convinced that most of us — even those who abhor the systems administrative aspects of computing as I do — need to know about connectivity and cells and gigabit packets (as well as the fact that ATM means something beyond the world of autotellers). The early chapters discuss building the networks that are able to execute data transmission at gigabit rates; the later ones concern protocols and applications. Partridge also has some interesting words on non-cell networking. It's good stuff and led me to reading *FDDI*.

Shah and Ramakrishnan have produced a handy, interesting volume on FDDI. They start out easy and manage to maintain the true textbook/tutorial approach, so that I understood most of their points. As it's obvious that Ethernet and token-ring won't have the burst capacity we need, understanding just what FDDI can (and can't) do is valuable.

Intelligent Behavior

Through the '70s and '80s I was interested in what people now call cognitive science. As a part of this I got involved in animal behavior and ethology. (I even got to participate in a seminar run by Konrad Lorenz in Salzburg in 1977). In 1983 I heard Marc Donner give a paper on walking; subsequently I published his paper on his walking language in *Computing Systems*. I thus have a long-standing interest in the relationship (there must be one) between animal behavior and robotics (which is not identical to that between cognition and artificial intelligence). McFarland and Bösser have now produced an interesting volume, *Intelligent Behavior in Animals and Robots*. Without wanting to go into a multipage discussion, M&B have decided that the Turing test focuses our attention excessively on language. They, instead, take an ecological approach, viewing the robot in its own milieu, much as Niko Tinbergen does in *The Animal in its World* (1972). M&B define intelligent behavior in terms of the behavioral outcome, not in terms of the mechanism that achieves it. In some ways, their work is quite antithetical to that of traditional AI research. This is an interesting and thought-provoking book, though irritating at times.

Stocking stuffers

I've mentioned both volume I (Software Design) and volume II (Software People) of P.J. Plauger's *Programming on Purpose*. Volume III (Software Technology) is now out. Do yourself a favor and read them. Get someone to buy all three for you for Christmas or whatever. As my Classics professor used to say, "Verb. sap."

Also, I think I ought to mention again, the best of the books mentioned earlier this year: Comer and Stevens' *Internetworking with TCP/IP*, vol. 3, (both BSD and TLI version), (Prentice Hall) is a genuine winner; so is Quarterman and Wilhelm's *UNIX, POSIX, and Open Systems* (Addison-Wesley). Harley Hahn's *Student's Guide to UNIX* (McGraw Hill) is certainly the standout among the recent spate of introductory volumes. A piece of Christmas marzipan goes to MIT Press for reissuing Hiltz and Turoff's *The Network Nation*. A second piece to Addison-Wesley, for Rich Stevens' explanation of TCP/IP protocols: *TCP/IP Illustrated*. And a whole box to O'Reilly for actually reducing the price on *The X Window System in a Nutshell* (originally published in 1990; 2nd ed. 1992) to under $10!

Note: ORA has published a third, revised edition of *Learning the UNIX Operating System*. This has been my favorite brief (under 100 page) introductory book ever since it appeared in 1986. It's better than ever.

Finally, though I most likely shouldn't make a comment, MIT Press has published the first USENIX monograph: Jim Waldo's *The Evolution of C++*. Most of these papers appeared in *Computing Systems* or were given at one of the C++ workshops/conferences. It's worthwhile to have them in one place at only $24.95.

Bryan Costales, Eric Allman, & Neil Rickert, *Sendmail*. O'Reilly & Associates, Inc., 1993. 83 pp. ISBN 1-56592-056-2. $32.95

Daniel P. Dern, *The Internet Guide for New Users*, McGraw-Hill, 1993. ISBN 0-07-016511-4. $27.95

Craig Partridge, *Gigabit Networking*. Addison-Wesley, 1993. 416pp. ISBN 0-201-56333-9. $46.25.

Amit Shah & G. Ramakrishnan, *FDDI: A High Speed Network*. Prentice Hall, 1993. 229pp. ISBN 0-13-308388-8.

David McFarland and Thomas Bösser, *Intelligent Behavior in Animals and Robots*. MIT Press, 1993. 308pp. ISBN 0-262-13293-1. $39.95.

P.J. Plauger, *Programming on Purpose: Essays on Software Technology*. Prentice Hall, 1993. 224pp. ISBN 0-13-328113-2. $27.00.

Grace Todino, John Strang, and Jerry Peck, *Learning the UNIX Operating System*. O'Reilly & Associates, Inc., 1993. 92pp. ISBN 1-56592-060-0. $9.

Words, Words, Words

***Jargon: An Informal Dictionary of Computer Terms* by Robin Williams. Peachpit Press, 1993. ISBN 0-938151-84-3. 676pp. $22**

***The New Hacker's Dictionary*. (2nd ed.) by Eric S. Raymond. MIT Press, 1993. ISBN 0-262-18154-1. 504pp. $14.95**

Reviewed by Peter H. Salus
<peter@usenix.org>

Because computing is an ever-developing field, its vocabulary is an ever-changing, ever-expanding one. It is thus appropriate for new vocabulary items to originate, for others to change their meanings, and for publishers to produce dictionaries. In October, two new ones occluded my mailbox: the second edition of Eric Raymond's *The New Hacker's Dictionary* and Robin Williams' *Jargon*. The bottom line of this brief essay is to recommend the first and wonder why the second was published, so if you aren't interested in "why," now you've got the answer.

Around 1970 I bought a copy of Chandor's *A Dictionary of Computers* and in 1977 I acquired its second edition (in a really handy Penguin book). On occasion, I still employ it. When I worked for IBM, I received their *Vocabulary forData Processing* .... I still look for things in the 1981 edition from time to time. In June 1982 the American Society for Testing and Materials held a conference in Toronto on Terminology, and the proceedings (*Standardization of Technical Terminology*; ASTM Special Technical Publication 806) were published in 1983. It is worth examining both new volumes in the light of the older ones.

But first, brief overviews. Williams' *Jargon* is nearly 700 pages of definitions ("A kilobyte is a unit of measure on a computer," p. 296); attempts

at humor ("LAN looks like an acronym that a board of directors spent a lot of money and time trying to create," p. 297); and drawings and cartoons. It attempts to be warm and fuzzy, but ends up being uninformative and cutesy. There is an entry for "Lady Lovelace" which mentions Charles Babbage as well as "Ada." There is no entry for either Babbage nor Ada. Many entries have drawings next to them indicating they are for Mac or PC users. Users of other systems must fend for themselves. (A graduate student I know, glancing at the book, suggested that Mac users were too dumb to find stuff without a logo to look for. I don't think this is necessarily true, but certainly Mac users are more accustomed to visual, not text oriented, lookups.)

The second edition of Raymond's book is something I have been looking forward to. It is now just over 500 pages, largely derived from the online JARGON files. There are over 200 new entries and, I'm told, over 175 of the old ones have been revised. There are still a few *bad things* (despite Heidi Stettner's statements that Biff did not bark at the mailman in 1980-83, the entry remains, p. 60), but this is a largely satisfactory volume. I am not a fan of Guy Steele's cartoons, but they certainly aren't bad or wrong.

Let me share a bit of comparative lexicography.

- In 1977, Chandor began the entry on *operating system* with:
  An operating system can be properly defined as those procedures which control the *resources* within a *data processing* installation [p. 295; the entry is nearly two pages long].

- Four years later, IBM's publication defined an OS as:
  Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management. Note: Although operating systems are primarily software, partial or complete hardware implementations are possible [p. 293].

- Williams' entry on operating system begins:
  The operating system (sometimes called the "OS") is the master control software that runs the computer itself. When you turn on your computer,the operating system is the first program that gets loaded into the memory of the machine. With the help of instructions built into the computer's ROMS or BIOS, the operating system sets up the means for your own programs to interact with the computer and its parts...
  and contains:
  Many *workstations* use the *Unix* operating system.
  and
  Other minicomputer and mainframe operating systems have names like VMS, VM, MVS, and even DOS, but these must be supplemented by "transaction processors" like CICS, TSO, and alphabet soup [p. 380].

- The "Hacker's" definition reads:
  The foundation software of a machine, of course; that which schedules tasks, allocates storage, and presents a default interface to the user between applications.... Hacker folklore has been shaped primarily by the UNIX, ITS, TOPS-10, TOPS-20/TWENEX, WAITS, CP/M, MS-DOS, and *Multics* operating systems (most importantly by ITS and *UNIX*) [pp. 312–313].

I thought that when things were referenced in a dictionary, they were listed. This is true of Chandor and of Raymond. While *memory*, BIOS, ROM (though out of alphabetical order), *workstation*, and *Unix* are listed in Williams, you can search in vain for the other items. I am not just picking nits here. I can find entries for *MS-DOS*, *MacBinary*, *Macintalk*, *Macintosh*, and *Macintosh Bible* in Williams, but not *emacs* or *TeX* or *telnet* or *ftp*, which are in Raymond.

I'm not sure the audience that Williams was aiming at— she specifically states that it's not just for the Mac or DOS crowd. But it isn't large. Outside of weight, you get less for more money than you would with Raymond.

DNS and Bind

***DNS and Bind* by Criket Liu & Paul Albitz. O'Reilly & Associates, Inc., 1993. 418 pages. ISBN1-56592-010-4. $29.95**

Reviewed by Peter Collinson
<pc@hillside.co.uk>

DNS & Bind is yet another excellent book from the O'Reilly stable. When you wander into some UNIX system administrators office you will expect to see a copy of this on their shelves soon, if not already. It's also relevant to anyone who wants to understand the glue that holds the Internet together. If you have no clue what the book talks about, then a small explanation is needed. Unless you have been asleep for the last ten years, you will now be familiar with the domain-style addresses for mail hosts and machines. DNS is the distributed database that allows your machine to translate from any domain address into a physical machine address on the Internet. This is done is a highly interactive manner, with the name server on your machine talking direct to name servers elsewhere.

Bind, for Berkeley Internet Name Domain software, is the widest used program suite that is written to support DNS specs. The visible piece of this will appear on your machine as a program called *named*. This program is a 'name server' and is sent name lookup requests from your local machines asking about the outside world. It will also get lookup requests about your site so other sites can talk to you.

The first three chapters of the book cover the basic philosophy behind the domain name scheme, domains in general and the operation of the DNS. Chapter 4 goes through all the whys and wherefores of setting up a domain name server on your machine. It involves the creation of several files holding information about your site. The chapter is really one large worked example, showing how a number of machines would be configured to support a name server service in a sensible way. Chapter 5 deals with how the DNS interacts with mail and grows the examples in Chapter 4.

Having set up your name server and got mail flowing into your site, you will then want make the other machines on your site access your name server. Chapter 6 goes into this in some detail. It talks about the peculiarities of some vendor implementations. For SunOS, it describes how NIS interacts with the name server.

Once you have things working, you will need to do periodic maintenance and checking. Chapter 7 is dedicated to this topic.

We are now sometime later in the life of your site. Your name server is overloaded and you want to find some way to spread the load. Chapter 8 discusses the various issues involved with this growth. It gives some tips on where name servers should be sited in your domain, discusses the load that name serving imposes on a machine. It goes onto talk about adding new name servers, providing redundancy. The chapter ends with what to do in a disaster, how to plan for it and what action to take in various situations ranging from bad to really bad.

Chapter 9 talks about how to delegate part of your domain to other parts of your organisation, this helps with load sharing but also fans out the administrative overhead into the hands of a local person.

Chapter 10 discusses *nslookup* , the program that you are likely to find on your machine to interrogate the DNS. The section looks at some common and less common tasks that you might wish to perform and tells you how to do them using *nslookup* . It gives points on how to recognise problems in your DNS setup.

Chapter 11 looks at the messages that BIND will dump into the *syslog*. There are verious levels of debugging output that you can get and this chapter shows you what the output should be and how to understand the information it is giving. Chapter 12 is a plethora of tips and information on how things go wrong, it includes common mistakes and many other problems. Chapter 13 is a programming guide to the routines that are used in the library to perform DNS lookup. If you want to write your own code to find Internet addresses, then this is the chapter for you.

Chapter 14 contains a number of tips and hints that don't fit easily elsewhere in the books framework. Finally the book ends with a number of appendicies: some message formats, how to compile BIND on a Sun, a list of top level domains, and the forms that you need to register your domain with the naming authority.

All in all, a good book and well worth the money. It contains a lot of good information, a wealth of tips and pointers to helpful software on the internet. I also like the nice touch that the now traditional fauna on the front cover is a cricket in a visual pun on one of the authors' names.

C++ Programming Style

C++ Programming Style by Tom Cargill. Addison-Wesley, 1992. ISBN 0-201-56365-7. 233 pages. $24.75

Reviewed by George W. Leach
<gleach@gte.com>

Alan Feuer once described the process of learning a programming language as comprising three basic steps [1]:

1. Understand the syntax
2. Understand the semantics of language constructs
3. Develop a style of programming that fits the language

This book provides the reader with the ability to tackle step 3 for C++. The approach taken to writing this book mimics that of Kernighan and Plauger's classic book, *The Elements of Programming Style.* [2]. Programs are culled from books, articles, and tutorials on C++ and improved upon. Then general rules are presented based upon the improvements. However, as the author points out in the Preface, this book does not concentrate upon lexical style or programming in the small issues The concentration is upon improving the design of classes and the relationships among them or programming in the large issues. The major topics covered by the chapters in this book are Abstraction, Consistency, Unnecessary Inheritance, Virtual Functions, Operator Overloading, Wrappers, Efficiency, A Case Study, Multiple Inheritance, and a Summary of the Rules. All of these topics are essential to constructing robust, efficient and reusable classes.

The first chapter discusses the topic of abstraction. It is typical of most of the book. An example program is presented that describes a computer and it's components using the class construct. The organization of the classes are incrementally improved and a series of rules drawn out of the experiences. Along the way, the reader is shown how modeling real world objects with classes does not necessarily result in the best implementation. Common abstractions are found and concentrated into base classes. The overuse of inheritance and virtual functions is explored and removed from the example. Differences in behavior and values are also investigated. As a result, the final program not only simplifies the design by reducing the number of classes and the size of the corresponding client program, but also provides additional flexibility to accommodate future changes.

Consistency is discussed with respect to the interface and implementation of classes. When building classes for use by others in client code, the class designer must take into account the expectations of the users. Consistent behavior of member functions, constructors, memory management and other aspects of classes is essential for avoiding problems.

Unnecessary inheritance builds upon the consistency material that precedes it by extending the same concepts into the area of the relationship among base classes and derived classes. An encapsulation hole caused by the manner in which inheritance was used is exposed and closed. Memory leaks are discovered. The question of what should be inherited is explored as well. Derived classes can inherit the interface to member functions, but change the implementation from the base class. Conversely, a derived class can inherit the implementation of the data and function members of a class, but change the interface. Or both could be inherited.

The appropriate distribution of data and code between base and derived classes is one of the main topics of the virtual functions chapters. Duplicate code is eliminated and the class structures simplified. Minimizing interactions between base and derived classes and the appropriate usage of virtual destructors are also covered.

Operator overloading is presented as both an extremely powerful and potentially dangerous language feature. There exists a tremendous potential for abuse that makes preprocessor abuse in C (e.g., `#define BEGIN {`) pale in comparison. All of the potential pitfalls of operator overloading are explored.

Wrapper classes, which interface to C functions are covered. When multiple instances of such a class are utilized in a program with an underlying C function that allows the calling program to access a static data element via a pointer, problems arise much like those associated with making a library thread-safe. Failure is also covered within the context of wrappers. One cannot depend upon a constructor when an underlying call, e.g., `fopen()`, can fail. Why? Because constructors don't return a value.

Efficiency is covered from a number of angles. The main emphasis is on separation of interface from implementation. In other words, the member functions and their arguments can remain the

same, but the internal implementation details of data members and the operations can change as long as the same effect is seen from the class users' perspective. This allows the class designer to apply efficiency tuning to a classes internal implementation and not require the users of the class to change their code to realize the benefits. Efficiency from the point of view of how instantiations of classes are often used by class users is also explored. The class designer must understand how to construct robust and efficient classes to anticipate potential uses. Temporary objects which are created through the use of operator overloading are also a major topic of this chapter.

A case study which presents a finite state machine program applies what has been presented in the first seven chapters to a larger example thus unifying all these concepts.

Multiple inheritance is one of Cargill's hot buttons. He has often advocated keeping this feature out of the language and can find no valid situation where its usage is warranted [3]. In this chapter he presents his case against this feature of C++.

The final chapter of the book presents a summary of the rules that were extracted from the improvements made to code throughout the book.

Each example in this book should be carefully studied by the reader prior to discovering how the author has improved upon it. Most of the chapters are approximately 20 pages in length and ideal for reading in a single sitting. Often the author challenges the reader to try and find further improvements. Take him up on it! You will get much more out of this book by taking your time with each chapter.

Cargill is one of the few people who are qualified to write such a book having written the first major application in the early 80s while at Bell Labs [4] and raising concerns about the utility of adding multiple inheritance to the language [3]. The style advocated in this book can best be summarized as a rather conservative approach to using the language features of C++. Certain features such as operator overloading, virtual functions and multiple inheritance are often not used appropriately. This book uncovers much misuse of language features and steers the programmer on the right path. Cargill has written an excellent book that should be on the book shelf of every serious programmer.

References

[1] Feuer, Alan R., *The C Puzzle Book*, Prentice-Hall, 1982.

[2] Kernighan, Brian W., and Plauger, P.J., *The Elements of Programming Style*, Second Edition, McGraw-Hill, 1978.

[3] Cargill, Tom, "Controversy: The Case Against Multiple Inheritance," in *Computing Systems*, Vol. 4 No. 2, Spring 1991, pp. 69-82.

[4] Cargill, T.A., "The Feel of Pi," *Proceedings* of the Winter USENIX Conference, Denver, CO, 1986.

Learning the Korn Shell

Learning the Korn Shell by Bill Rosenblatt. O'Reilly & Associates, Inc., 1993. 338 pages. ISBN 1-56592-054-6.

Reviewed by Bob Birss
<*bob.birss@sun.com*>

Bill Rosenblatt tells us that this book "is designed to appeal most closely to casual UNIX users who are just above the 'raw beginner' level." I picked it up because I'd decided to convert to Korn shell from C shell, which I'd been using for years. Chapters such as "Korn Shell Administration" and "Debugging Shell Programs" looked like they would be of use, even though I had a copy of Morris I. Bolsky and David G. Korn's *The Kornshell Command and Programming Language* (Prentice Hall, 1989). What particularly attracted me was *kshdb*, a "full-blown" shell script debugger—with breakpoints, break conditions, stepping, and execution trace toggling. The complete source code is given, as are several sources for getting an electronic version.

I read through Rosenblatt's book, spending varying amounts of time in its various parts. I found it particularly useful that Rosenblatt points out differences between the Korn and C and Bourne shells as he moves through his discussion, especially in his first chapter, "Korn Shell Basics."

However, when I started to convert my *.login* and *.cshrc* files, I wished that the book had more "reference material." It would be useful — to someone converting from another shell as well as to the new user —to see something similar to the sample *.profile* and environment files given in Bolsky and Korn's book. There is Appendix B, "Reference Lists," which is intended for reference purpose, and there is a full index and are many examples in the text. But numerous times I found myself digging around in the book trying to find the answers to my questions. For example, though there is a long section on "Shell Variables" in the chapter "Customizing Your Environment" and a list of "Built-In Shell Variables" in Appendix B, I found it unclear whether to explicitly export ENV. Rosenblatt seems to suggest that it's "built in," which he says should make it automatically available to subshells. Moreover, in his sample *.profile* on page 82, Rosenblatt doesn't export it. As it turns out, the implementation of *ksh* on my system requires exporting ENV — which I had to discover through trial-and-error.

Rosenblatt's discussion of *kshdb* is undeniably useful—it illustrates many aspects of Korn shell programming, and it should facilitate debugging shell scripts. However, initial trial provided disappointing results:

- Invoking *kshdb* with no argument produces the message `Cannot read`, which baffled me for a moment —until I looked at the code and discovered that *kshdb* requires an argument, but tests if its argument is the name of a readable file. Unfortunately, a null argument isn't such a name and so *kshdb* produces an unreadable message.
- *kshdb* is not capable of debugging a script on my system. Loading any Korn shell script —including *kshdb* itself, as well as the example scripts provided with it — produces a "bad trap" on my system and *kshdb* exits.

I cannot recommend this book to its intended audience. "Casual UNIX users who are just above the 'raw beginner' level" would likely be confused or mislead by much in the book. The advanced user would, however, likely find a good deal of interest —if he or she were prepared for some trial and error and some shell script programming and debugging.

High Performance Computing

High Performance Computing by Kevin Dowd, O'Reilly & Associates, Inc. 1993. 371pp. ISBN 1-56592-032-5. $25.95

Reviewed by Vern Paxson

Writing efficient programs can prove trickier than we might at first expect, particularly with today's hardware platforms that, if properly catered to, offer great performance potential. Often, knowledge of how to write and tune programs for high performance comes only in the form of lore passed along from one programmer to another. Kevin Dowd's *High Performance Computing* is a meaty collection of such lore, rounded out with overviews of modern RISC architectures, optimizing compilers, and parallel computing. The first two add valuable context to the general discussion; the third provides a look at what for many of us will be tomorrow's computing environment.

The book limits its scope to primarily scientific computing with an emphasis on FORTRAN (though C gets some attention) running on UNIX platforms. As the author states in the preface, this focus means skipping other areas such as I/O, graphics, and algorithm design. But even with a narrowed viewpoint, he covers a rich set of topics, ranging from modern CPU and memory system architecture, to program timing and profiling, to optimizing loops and memory references, to proper benchmarking. Each topic is treated with considerable energy and expertise, written in a pleasant, homey style, and includes well-chosen examples.

Indeed, the book suffers from sometimes covering material with a degree of detail surpassing its value to the central discussion. For example, the reader really doesn't need to know the details of microcoding, nor how compilers turn basic blocks into DAGs, nor how to give program variables mnemonic names. I found myself wishing that the energy that went into these sections had instead been used to construct a clearer overview of how all the pieces fit together. The book begins with 80 pages discussing RISC architectures and optimizing compilers, but readers whose primary interest is in speeding up their large FOR-

TRAN programs have not been given any context with which to assess this material's relevance to what will follow. Such readers might actually be better off reading the middle of the book before the beginning.

The book is primarily for the efficiency newcomer; more seasoned programmers have already absorbed much of the lore. With this in mind, the book would benefit from underscoring some of the fundamentals of writing high-performance programs: algorithms can make a huge difference, don't spend energy on performance where it doesn't matter (hence the vital importance of profiling), and beware how undisciplined performance tweaking degrades a program's maintainability. The book discusses each of these points, but not in a way that emphasizes them as fundamentals.

One criticism I have is that the book lacks summaries and checklists. While the author states that this is his intention, the material is thick enough and intertwined enough that many readers may find it difficult to know where to begin for their particular task at hand. The one checklist included in the book — for benchmarking — proves a valuable quick reference. More such would bolster the book's ease of use.

The book's index is adequate but not inspiring (no entry for "locality of reference," for example), and there are no references to other books such as good algorithm or architecture texts. But there is no denying the quantity of well-presented, valuable information. Given all of the above, I recommend the book strongly for those new to high-performance computing; those with more experience are still likely to find some fresh notions and techniques.

Learning the UNIX Operating System

***Learning the UNIX Operating System (3rd Edition)* by Grace Todino, John Strang, and Jerry Peek. O'Reilly & Associates, Inc. 1993. ISBN 1-56592-060-0. 108pp. $9.95**

Reviewed by Betsy Gillies
<asl@symcom.math.uiuc.edu>

For someone who has only encountered computers via a word processor, this is just the ticket. UNIX can be a bit daunting whithout a guru nearby, and this slim volume is designed to make you feel very comfortable right from the beginning. For more experienced computer types, it may be a bit too simple, but I have noticed that most writers in the field have long since forgotten what it is like to be a babe-in-the woods and seem to take for granted that the new user can make his/her way through all kinds of problems. This book doesn't do that.

I found the section on mail clear and easy to understand, although I wish it had to introduced the user to the *alias* facility, so useful in avoiding the many possibilities for error in correctly remembering and typing email addresses. Also, it would have been a kindness to tell the reader that a saved email file is the same as any other UNIX file and can be handled as such, using *vi* or *emacs* outside the mail system.

The section on file management will be very helpful for a novice.

I have shared this book with graduate student Adam Lewenberg at the University of Illinois, who now plans to use it for several short courses he teaches about email, UNIX, and Windows. These courses are taken by mathematics faculty members, students, and office staff who wish to use mail and have some basic familiarity with the UNIX system.

Publications from John Wiley & Sons, Inc.

Member Number: _____

All USENIX members receive a 15% discount on the following Wiley publications

Introduction to Client/Server Systems
Paul Renaud
1-57774-X $34.95 ***member price: $29.71***
# of Copies: _____

Portable UNIX
Douglas Topham
1-57926-2 $14.95 ***member price: $12.71***
# of Copies: _____

The UNIX Command Reference Guide
Kaare Christian
1-85580-4 $32.95 ***member price: $28.01***
# of Copies: _____

C++ For Programmers
Leendert Ammeraal
1-93011-3 $32.95 ***member price: $28.01***
# of Copies: _____

Obfuscated C and Other Mysteries
Don Libes
1-57805-3 $39.95 ***member price: $33.96***
# of Copies: _____

Adventures in UNIX Network Applications Programming
Bill Rieken
1-52858-7 $34.95 ***member price: $29.71***
# of Copies: _____

UNIX, Self Teaching Guide
George Leach
1-57924-6 $19.95 ***member price: $16.96***
# of Copies: _____

UNIX Shell Programming, Second Edition
Lowell Jay Arhur
1-51821-2 $29.95 ***member price: $25.46***
# of Copies: _____

Object Oriented Programming withTurbo C++
Keith Weiskamp
1-52466-2 $24.95 ***member price: $21.21***
# of Copies: _____

Berkeley UNIX
A Simple & Comprehensive Guide
James Wilson
1-61582-X $37.95 ***member price: $32.26***
# of Copies: _____

U.S. orders only

☐ Payment enclosed, plus sales tax
☐ VISA ☐ Mastercard
☐ American Express
Card No. _____

Name _____
Firm _____
Address _____
City/State/Zip _____
Signature _____
(order invalid unless signed)

Please mail or fax orders to the following address:

John Wiley & Sons, Inc.
605 Third Avenue
New York, NY 10158
Attn: Karen Cooper

Phone: (212) 850-6789
FAX: (212) 850-6142

WILEY

McGraw–Hill Books
20% Discount for USENIX Members

Qty

Open Systems Interconnection Its Architecture and Protocols, Revised Edition..... ☐

Bijendra N. Jain & Ashok K. Agrawala 0-07-032385-2 hardcover
retail: $50.00 member price: $40.00

A Students Guide to UNIX..... ☐

Harley Hahn 0-07-025511-3 paperback
retail: $25.75 member price: $20.60

Data Network Design Packet Switching, Frame Relay, 802.6/DQDB..... ☐

Darren L. Spohn 0-07-060360-X hardcover
retail: $59.00 member price: $47.20

Migrating to Open Systems: Taming the Tiger..... ☐

Daniel R. Perley, 0-07-707778-4 hardcover
retail: $30.00 member price: 24.00

I am a member of the USENIX Association. Please send me the books I have indicated at the member special rate. I have added $3.00 postage and handling for the first book ordered, $1.00 for each additional book, plus my local sales tax.

Total price of books _____
Sales tax _____
Postage & Handling _____
Total enclosed _____

USENIX Membership #_____

____ Check or money order is enclosed - payable to McGraw-Hill, Inc.

____ Charge my ____ Visa ____ Mastercard ____ Discover ____ Amex

Account #_____

Expiration Date_____

Signature_____

Shipping Information (7/93 - 03US001)

Name _____

Address _____

_____

Daytime Phone # _____

Send orders to:

McGraw-Hill, Inc.
Attn: Rosa Perez
11 West 19th Street - 4th Floor
New York, New York 10011

If I am not completely satisfied, I will return the book(s) within 15 days or a full refund or credit.
Satisfaction unconditionally guaranteed. Prices subject to change without notice. We can only accept orders within the continental USA.

Announcements/Calls for Upcoming Events

On the following pages are Announcements and Calls for Papers (CFPs) for upcoming USENIX events. Watch *comp.org.usenix* for regular postings of updates, student grant applications, registration, and hotel information.

Computing Systems

On the verge of beginning its seventh year of publication, *Computing Systems* has become one of the most important and frequently-cited journals in the field of advanced computing.

Rapidly achieving truly worldwide status, *Computing Systems* has published work by researchers in Australia, Canada, Finland, France, Italy, the Netherlands, Switzerland, and the UK, as well as North America.

The quarterly journal is dedicated to the analysis and understanding of the theory, design, art, engineering, and implementation of advanced computing systems, with an emphasis on systems inspired or influenced by the UNIX tradition. The journal's content concerns operating systems, architecture, networking, programming languages, and sophisticated applications.

Manuscripts should be sent to <*journal@usenix.org*>. Alternatively, they may be sent to the postal address below, together with a cover sheet listing the authors' names and telephone numbers, the title of the article, and a 75–300 word summary.

All submissions will be reviewed by at least two members of the Editorial Advisory Board as well as theEditor-in-Chief. Comments from reviewers will be transmitted to authors anonymously, unless the reviewer has indicated that his or her remarks can be attributed.

Editorial correspondence should addressed to:

Email: <*journal@usenix.org*> or
Managing Editor
Computing Systems
2560 Ninth Street, Suite 215
Berkeley, CA 94710

DUAL-TRACK TECHNICAL SESSIONS

MONDAY–WEDNESDAY, JANUARY 17–19, 9:00 AM–5:30 PM

☛ **KEYNOTE:** John Perry Barlow will speak on recent developments in the national information infrastructure, telecommunications regulation, cryptography, globalization of the Net, intellectual property, and, generally, of the settlement of Cyberspace. In 1990, Mr. Barlow and Mitchell Kapor co-founded the Electronic Frontier Foundation, and he currently serves as chair of its executive committee.

- ☛ Presentations of papers refereed by the Program Committee and published in the Conference Proceedings.
- ☛ A parallel track of Invited Talks provide overviews and practical commentary on UNIX on Wall Street, C++ and Garbage Collection, Automatically Generating 99% of the Compiler, Facsimile Transmission – Now and in the Future, State-of-the-Art Video Compression, Cons as well as Pros of OO Programming.
- ☛ Highlights from recent USENIX and other Symposia, Work-in-Progress Reports, and The Guru is IN.

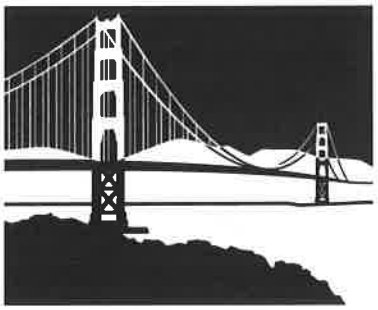
SPECIAL HOTEL AND AIRFARE RATES

- ☛ USENIX attendees receive special rates at our conference headquarters, the San Francisco Hilton (Tollfree: 1-800-HILTONS–U.S. or Canada), and at the nearby King George Hotel (Tollfree: 1-800-288-6005–U.S. or Canada).
- ☛ Reserve at the hotel of your choice before December 27 for special rate. **BE SURE TO MENTION YOU ARE ATTENDING USENIX.**
- ☛ Save when flying American OR United Airlines. Savings available only through JNR, Inc. (Tollfree: 1-800-343-4546–U.S. or 1-714-476-2788), who will help you plan your flights on all airlines, one-call changes and refunds, boarding passes, and two free drink/headset coupons with each ticket.

TO RECEIVE COMPLETE CONFERENCE INFORMATION

PLEASE CONTACT:
USENIX CONFERENCE OFFICE
1-714-588-8649
FAX 1-714-588-9706
E-MAIL: conference@usenix.org

U S E N I X
WINTER 1994
CONFERENCE



JAN. 17–21, 1994
SAN FRANCISCO
CALIFORNIA

IMPORTANT CONFERENCE DATES

REGISTRATION SAVINGS DEADLINE
TUESDAY, DECEMBER 27, 1993

HOTEL DISCOUNT DEADLINE
TUESDAY, DECEMBER 27, 1993

TECHNICAL SESSIONS
MONDAY–WEDNESDAY,
JANUARY 17–19, 1994

TUTORIALS
THURSDAY AND FRIDAY,
JANUARY 20 AND 21, 1994

PRODUCT DEMONSTRATIONS & VENDOR DISPLAY
TUESDAY, JANUARY 18, 1994
NOON – 5:00 PM
WEDNESDAY, JANUARY 19, 1994
10:00 AM–2:00 PM

PRE-REGISTER BY DECEMBER 27 TO SAVE UP TO $100

FAST-PACED, PRACTICAL TUTORIALS

THURSDAY & FRIDAY, JANUARY 20 AND 21, 9:00 AM–5:00 PM

Choose among nineteen, fast-paced, full-day tutorials. USENIX's well-respected tutorial program emphasizes immediately useful information. Instructors are experienced teachers as well as experts in their topics. At San Francisco, you may attend one or two full-day tutorials in the following areas:

- ☛ Essential UNIX Programming
- ☛ UNIX Network Programming
- ☛ Getting the Most from UNIX Power Tools
- ☛ State-of-the-Art System Administration
- ☛ Sendmail Inside and Out
- ☛ Intro to Threads, POSIX PThreads, and OSF/DCE Threads
- ☛ Windows NT – Architectural Overview
- ☛ Windows NT – Developing Client-Server Applications
- ☛ CHORUS and SVR4 UNIX
- ☛ Porting to Solaris 2.x
- ☛ OSF's Distributed Computing Environment
- ☛ Client-Server Development with DCE/RPC
- ☛ How Networks Work – Architectural Implications
- ☛ TCL and TK: New Approach to X11 and GUI Programming
- ☛ Distributed Object Computing with CORBA
- ☛ Achieving Security in an Internet Environment
- ☛ Kerberos Approach to Network Security
- ☛ The Law and the Internet

EVERYONE'S INVITED – ADMISSION IS FREE
PRODUCT DEMONSTRATIONS & VENDOR DISPLAY

TUESDAY, JAN. 18, NOON–5:00 PM &
WEDNESDAY, JAN. 19, 10:00 AM–2:00 PM

In scheduled 30-minute presentations, vendors will demonstrate the technological innovations underlying their products. In the USENIX Vendor Display emphasis is on serious answers from technically savvy vendor representatives, so you'll know how something will work with what you've already got. Plus, you can review the newest releases from technical publishers.

For free admission or vendor information, contact: Cynthia Deno 1-408-335-9445, E-mail: cynthia@usenix.org

USENIX The UNIX and Advanced Computing Systems Professional and Technical Association

SECURITY & SYSTEM ADMINISTRATION MANAGEMENT SEMINARS

at the UniForum Conference

MARCH 21 & 22, 1994

MOSCONE CONVENTION CENTER
SAN FRANCISCO, CALIFORNIA

Sponsored by

USENIX and SAGE THE SYSTEM ADMINISTRATORS GUILD

ORGANIZING COMMITTEE:

Paul Evans, SRI International, Ellie Young, USENIX, and Daniel Klein, USENIX

TO REGISTER:

You may choose to attend one or both days of either of the two seminars: One day $295; two days: $500 (before February 18; fee includes lunch.) Call 1-800-225-4698 (in U.S.A.) or 1-508-879-6700 to request UniForum '94 registration form.

WHO SHOULD ATTEND?

UNIX system managers and system administrators charged with maintaining security or with meeting their organization's system administration needs. You may choose to attend one or both days of either of the two seminars. The 1994 UniForum Conference is the first to offer seminars sponsored by the USENIX Association and SAGE, the System Administrators Guild.

These intensive seminars explore effective management techniques in two critical areas: protecting security and managing computing services. The first of the seminars examines immediate and long-term policies with which to combat security threats to your computing environment and, specifically, to the X-Window System. The second seminar provides insights and policies to achieve effective computing services management, while building a crack team of system administrators.

Seminar 1: SECURITY

If you are charged with protecting your organization's systems and networks against intrusion by malicious or mischievous parties (from both within and without), this session is for you. Learn how to recognize fundamental security risks. You'll also get an overview of the security features (and failings) available under UNIX, and the implications of security measures for interoperability and user-friendliness.

MONDAY • MARCH 21, 1994 • 9:00 AM–5:00 PM

UNIX System and Network Security

Rob Kolstad and Bill Rieken

The first half of the session covers issues such as file and directory protection, how to secure modems and multiple types of networks (LANs and WANs). You'll also learn about state-of-the-art security techniques such as Kerberos systems and network firewalls. In the afternoon, the seminar turns to the specifics of maintaining security in a UNIX environment. System accounting commands, the crypt command, passwords, file protections, and the mount command are all demonstrated in the context of system security. A custom security audit daemon is shown, along with sample output, to help you monitor your system. This half of the seminar also covers some details of UUCP security, SUID programming, Trojan Horses, and security audit and logging.

TUESDAY • MARCH 22, 1994 • 9:00 AM–5:00 PM

Security and the X Window System

Jeremy Epstein and Rita Pascale

What are the risks to security, and available solutions, posed by the X Window System? This seminar provides a basic introduction to both X and security – no advance in depth knowledge is needed. As the X Window System increases in popularity, so does concern about its security. Because X is an open, resource-sharing system, security measures are not easily retrofitted without damaging interoperability, and there are no quick and simple answers to security risks. Learn about new security-enhanced X systems being introduced by some vendors, and their implications for programmers and other users. Other topics covered include threats, current technologies, authentication, access controls, auditing, privilege, and denial of service. Use of authentication mechanisms is described in detail, including *xhost*, MIT magic cookies, Sun's Secure RPC, and Kerberos. Vendor-specific extensions to X for access control and privilege are presented. Alternate architectures are described for multi-level secure X-Systems.

Seminar 2: MANAGING AND DECISION-MAKING FOR THE SYSTEM ADMINISTRATION PROCESS

This two day, in-depth seminar teaches the how-tos of keeping computer systems on-line, of providing efficient computer services, and of building a crack team of system administrators. The format is a series of one- to two-hour long presentations in which veteran computing services managers share their experience. These presentations are designed to enhance your success as a manager of computing services and of a system administration staff.

MONDAY • MARCH 21, 1994 • 9:00 AM–5:00 PM

System Administrator Hiring and Job Performance

Tina Darmohray

Meet the challenges of building a highly effective system administration team at your site. As a manager of a growing staff of system administrators, this session provides you with a sophisticated understanding of what to look for when hiring, what is a good training program for your site, and, then, what kind of job performance you should expect.

System Administrators from the Manager's Perspective

William E. Howell

System administration is an exciting, but rapidly changing and ill-defined field. Learn to cope with the unique challenges of system administration management such as: growing out-of-touch with new technology; managing people who are doing work you've never done; and dealing with a poorly-planned management structure.

Managing for Ethical and Legal Computer Use

Rob Kolstad

Dealing with large user communities leads to new problems in data collection, software licensing, security, and ethics. We review the pros and cons of dozens of policies on issues ranging from e-mail privacy to use of copyrighted software. Get experience-based, common-sensical approaches to creating an organization-wide policy for computer use in your environment.

Making Sure Free Software is a Bargain

Rob Kolstad

Network software repositories contain a wealth of interesting and valuable software, such as very high level languages, compilers, source control systems, network protocols – available for no charge. In this section, we discuss a number of the free or public domain packages at length and show you how to find specific software using one of the many archival servers on the net. We also discuss the caveats of using this type of software and explain some of the drawbacks of various policies that may come with it.

TUESDAY • MARCH 22, 1994 • 9:00 AM–5:00 PM

Preventing Backup Disasters

Elizabeth Zwicky

No collection of system administration disasters is complete without a "no backup" disaster story. This session helps you avoid personal experience of what can be a true tragedy. The backup system that is well suited to one site may be completely inappropriate to another. This presentation discusses the issues involved in selecting a backup system and setting up a backup schedule uniquely suited to your site. We offer specific guidelines for choosing among the many available backup systems.

Making Do with a Limited System Administration Staff

Paul Evans

The industry trend to distributed computing is producing sites where the machine-to-system administrator ratio is frequently greater than 100 to 1. Learn how to live with this ratio by seeking strong management backing, using creative administration management, and, understanding where to focus system administration efforts. Learn techniques for implementing the time-saving software and hardware solutions available to tackle the special problems of running a large site with limited staff.

What Do System Administrators Really Do?

Elizabeth Zwicky

Managers, vendors, and system administrator are all frequently frustrated by what seems to be a massive communications gap. System administrators are often violently opposed to vendor's best-intentioned attempts to provide them with tools. They may be engaged in mutually puzzling dialogues with management on topics like staffing, appropriate goals, and performance measurements. This section discusses the causes and symptoms of this situation. We use a "day in the life" of a typical system administrator to contrast what system administrators really do with common management/user/vendor (mis)perceptions.

The Canons of Computer Services Management

William E. Howell

This section explores a variety of management approaches and styles and shares new ideas and views that will improve your management effectiveness. We consider your spheres of influence as a manager: your management, the computing environment you maintain, your vendors, your employees, your user community, and your organization's projects. For each of these we present canons or key principles that will help you be a stronger, more effective and more efficient manager.

Recognized Experts Sharing Their Experience

Dr. Rob Kolstad is Program Manager for Berkeley Software Design, Inc. Previously, he was staff engineer at Sun Microsystems' Rocky Mountain Technology Center in Colorado Springs, Colorado where he led development of Sun's new Backup Copilot product. Rob is editor of *;login:*, the newsletter of the USENIX Association.

Bill Rieken has taught at the University of Wisconsin and Southern Illinois University. A principal founder of .sh Consulting in Santa Clara, he publishes seventeen UNIX training books for the courses he teaches, and is co-author of *"Adventures in UNIX Network Applications Programming"* (Wiley, 1992)

Rita Pascale is a researcher in highly trusted distributed systems at ORA Corporation. Until recently, she worked on trusted windowing systems at TRW. She has written four papers on X and security.

Jeremy Epstein is Chief Engineer at Cordant, responsible for design and development of the trusted Novell NetWare product. Previously, he was a researcher in highly trusted windowing systems at TRW, and has written nine papers on the subject of X and security.

Elizabeth D. Zwicky is a Senior System Administrator at SRI International and a board member of SAGE. She has been involved in system administration since a Sun 2 was an exciting computer and 100 machines was a big network, and has published or presented dozens of papers on system administration topics.

Paul Evans is currently a Senior System Administrator at SRI International, where he and his colleagues manage a network of about 200 workstations on several dozen networks. He spent three years at MasPar Computer Corporation, where he singlehandedly ran a network of 150 workstations. He is a founding member of BayLISA and SAGE, and has served on the board of directors of both organizations.

Tina Darmohray is the Lead for the UNIX System Administration Team at Lawrence Livermore National Laboratory, where her group has responsibility for over 1,000 machines. In 1990, she installed the first firewall at LLNL and has since consulted with a number of sites in the Bay Area.

William E. (Bill) Howell is the Director of Computer Services for the Graduate Department of Computer Science at The University of North Carolina at Chapel Hill. In Bill's decade at UNC, he has overseen the expansion of the facility to more than 450 systems, 240 gigabytes of disk space, and a comprehensive high speed network for video, data, and voice. An experienced, college-level instructor, Bill also consults on project management, computer security, and management of computer service organizations.

CALL FOR PARTICIPATION

USENIX UNIX APPLICATIONS DEVELOPMENT SYMPOSIUM

APRIL 25–28, 1994
MARRIOTT HOTEL
TORONTO,
ONTARIO; CANADA

IMPORTANT DATES

DATES FOR REFEREED PAPER SUBMISSIONS:

- **Extended Abstracts Due:** January 10, 1994
- **Notifications to Authors:** January 26, 1994
- **Final Papers Due:** March 11, 1994

OTHER IMPORTANT DATES:

- **Pre-Registration Materials available:** Mid-February, 1994
- **Tutorial Program:** Monday & Tuesday, April 25 & 26
- **Technical Sessions:** Wednesday & Thursday, April 27 & 28
- **Birds-Of-a-Feather Sessions:** Monday–Thursday evenings
- **USENIX Reception:** Wednesday evening, April 27

CO-SPONSORED BY THE USENIX ASSOCIATION AND UNIFORUM CANADA

One of the major uses of UNIX is the support, development, and execution of applications which ultimately serve as tools for end-users. In addition, the current trend of downsizing major applications from monolithic data-center environments to less expensive, distributed workstations and client-server computing environments affords UNIX a serious position in the commercial marketplace. Because UNIX has become a viable commercial alternative, vendors are now porting and developing code for scientific and business applications which in the past have been the province of contributed code. Consequently, more and more computing and information systems professionals are encountering UNIX when developing and maintaining applications.

The purpose of the **UNIX Applications Development Symposium** is to expose the challenges of building and maintaining applications on UNIX platforms, to discuss solutions and experiences, and to explore existing practice and technique. Computing professionals who have long viewed UNIX as the program development platform of choice, as well as professionals new to the UNIX environment, will learn of helpful tools, novel approaches, and what ***not*** to do when developing for or porting an application to the UNIX environment.

The symposium will feature technical papers, invited talks, panel discussions, and tutorials on all aspects of designing, building, testing, debugging, and maintaining applications within and for the UNIX environment. There will be ample opportunity to meet your peers and make contact with others with similar interests.

The **UNIX Applications Development Symposium** will provide valuable information to designers, programmers, and managers who plan to port existing applications into the UNIX environment or move development and maintenance teams from various proprietary environments to UNIX.

TUTORIAL PROGRAM

The two, day-long tutorials are targeted to programmers and managers interested in developing applications in, and products for, the UNIX environment. Each is led by an experienced instructor who is an expert in his topic. The Monday tutorial by Richard Stevens covers the use of the UNIX environment to develop applications. The tutorial on Tuesday, presented by Rob Kolstad, covers design and implementation issues regarding effective use by an application of the UNIX environment.

INVITED TALKS AND PANEL SESSIONS

As part of the technical sessions, invited talks provide introductory and advanced information about a variety of interesting topics, such as using standard UNIX tools and employing specialized applications. We welcome suggestions for topics, as well as request proposals for particular talks. You are encouraged to direct a proposal to the program chair. State the main focus, include an outline, and emphasize why the topic of the talk is of general interest.

Panel sessions on technical issues are welcome. Persons interested in participating in panel discussions should also contact the program chair.

WORKS-IN-PROGRESS REPORTS

These reports provide researchers, developers, and implementors with ten minutes to speak on current work and receive valuable feedback. Present your interim results, novel approaches, or newly-completed work. Schedule your report in advance or on-site. Inquiries about WIPs should be directed to the program chair.

SUGGESTED TOPICS

- **Graphical User Interfaces** – The X Window System. User Interface Design and Standards. Open Look, Motif, and NeWS. Style guides and toolkits. Importance of consistency and ease of use.

(continued on reverse side)

Program Committee

- **Program Chair:**
  Jim Duncan, *Math Depart., The Pennsylvania State University*
- **Program Vice-Chair:**
  Greg Woods, *GAW Consulting*

Dan Heller, *Z-Code Software, Inc.*

Rob Kolstad, *Berkeley Software Design, Inc.*

Evan Leibovitch, *Sound Software*

Peter Renzland, *Ontario Government*

Dan Tomlinson, *Compusoft*

Elizabeth Zwicky, *SRI International Inc.*

Dates For Refereed Paper Submissions

- **Extended Abstracts Due:**
  January 10, 1994
- **Notifications to Authors:**
  January 26, 1994
- **Final Papers Due:**
  March 11, 1994

For Program and Registration Information

Materials containing all details of the technical and tutorial programs, conference registration, and hotel and airline discounts and reservation information will be mailed in mid-February 1994. If you wish to receive the registration materials, please contact:

- **USENIX Conference Office**
  22672 Lambert St., Suite 613
  Lake Forest CA USA 92630
  +1 (714) 588-8649
  FAX: +1 (714) 588-9706
  E-mail: conference@usenix.org

The UNIX and Advanced Computing Systems Professional and Technical Association.

- **Porting Issues** – Issues surrounding the tasks of porting an existing application to UNIX, as well as issues of making UNIX applications portable to other architectures and other platforms. POSIX compliance.
- **Networking – Client/Server design issues.** How and where to separate the functions of clients and servers. Novel paradigms. The impact of mobile computing on application design and testing. The impact of network design or selection on application development and performance.
- **CASE and Project Management** – Using UNIX tools and environment to support code development and project management. Notable gains and losses. Modifications and adaptations to well-known techniques.
- **Operating System Issues** – Adapting to limitations or benefits of various hardware platforms and operating systems. POSIX and COSE.
- **Security** – The impact of security features. Schemes for maintaining security within an application. Client/server issues. Encryption schemes. Issues affecting integrity, reliability, and non-repudiation. Using Kerberos and other third-party systems in applications.
- **Transaction Processing** – Implementing distributed transaction processing for UNIX applications. Performance and scaling issues.
- **Distributed Applications** – How do you make the best use of existing UNIX functionality to build UNIX applications? Novel solutions. Client/server considerations.
- **Object Oriented Programming** – Productivity, languages, techniques, case studies. Experiences using C++, Eiffel, or other languages in code development.
- **Internetworking** – Effects on application design and support. Interesting or useful development platforms. Portability issues. Appropriate use. Advantages and disadvantages of various network architectures.
- **Delivering and Installing Applications** – Best methods. Superuser requirements. Licenses and license administration. Software piracy. Preventing worms and viruses. How to do updates effectively, economically, and securely.
- **Testing and Certification** – The impact of compliance. Experiences coding for and meeting compliance with various standards. Applications and POSIX.1 Conformance Testing.
- **Application Standards** – What are ABI, API, and ANDF? Selection criteria and impact on application design and development.

Submission Details

Papers may feature real-life experiences, as well as research topics. Both case-study and technical papers will be accepted. Case studies should describe existing systems and include implementation details; performance data is strongly encouraged.

A submission must be in the form of an extended abstract (1500–2500 words, 3–5 pages in length). The extended abstract should represent your paper in short form. It should demonstrate that you have a real project, that you are familiar with the work in your area, and that you can clearly explain yourself.

Papers will be judged on technical merit, relevance to the theme, and suitability for presentation. Software and hardware developers who wish to share their experiences, innovative solutions, and techniques are encouraged to submit papers.

Please submit one copy of an extended abstract (e-mail preferred) via:

- **E-mail:** <app-dev-sub@math.psu.edu>
- **FAX:** +1 (814) 865-3735 to Jim Duncan re: USENIX App Dev 94
- **Postal mail:** Jim Duncan
  USENIX App Dev 94
  Math Depart., Penn State University
  218 McAllister Building
  University Park PA USA 16802

Please refer to **"USENIX App Dev 94"** on all FAXes and postal mail.
Please direct inquiries regarding the symposium to <jim@math.psu.edu>.

USENIX SUMMER 1994 TECHNICAL CONFERENCE

JUNE 6–10, 1994
BOSTON MARRIOTT
COPLEY PLACE HOTEL,
BOSTON,
MASSACHUSETTS

COME JOIN US IN CELEBRATING THE 25TH ANNIVERSARY OF UNIX!

IMPORTANT DATES

Dates for Refereed Paper Submissions:

- Submissions due: January 28, 1994
- Authors notified by: February 21, 1994
- Camera-ready papers due: April 8, 1994

Registration Materials Available:

- Mid-March, 1994

CONFERENCE SCHEDULE IN BRIEF

- **Tutorial Program:**
  Monday and Tuesday June 6–7, 1994
- Technical Sessions:
  Wed. through Fri., June 8–10, 1994
- **Birds-of-a-Feather Sessions:**
  Tues. through Thurs., June 7–9, 1994
- **USENIX Reception:**
  Thursday evening, June 9, 1994
- **Vendor Display:**
  Wed. and Thurs., June 8–9, 1994

ANNOUNCEMENT & CALL FOR SUBMISSIONS

As always at a USENIX conference, we are interested in papers describing new and interesting developments in and around open systems, including networking, new languages, tools, and software engineering. Our traditional focus on UNIX remains, but we also encourage submissions about systems other than UNIX, that have a significant impact on the world of open systems.

Our emphasis this time is on UNIX in the real world: developing portable applications, running businesses on UNIX, using UNIX as an educational platform. Share the trials and tribulations of migrating complex applications into a UNIX environment. Describe the network protocols necessary to make the networking superhighway a reality. Or, take a foray into the future of computing:

- What to do when your toaster runs UNIX: Does cat still work?
- What should we do now to prepare for the 21st century?
- Of Mice and Pen: Human-computer interaction
- UNIX OS for the Masses
- Terabytes a Day: What's a disk to do?

TUTORIAL PROGRAM

On Monday and Tuesday of the conference, you can attend intensive and practical tutorials in topics essential to anyone using or developing UNIX and UNIX-like operating systems, X windows, networks, advanced programming languages, and related technologies. The USENIX Association's well-respected program offers you both introductory and advanced courses, presented by skilled teachers who are hands-on experts in their topic areas.

At Boston, USENIX will offer two full days of tutorials on topics such as:

- Introductory and advanced system administration
- Distributed computing: DCE, DFS, RPC, COBRA
- Kernel internals: SVR4, Chorus, Windows NT
- Systems programming tools and program development
- Systems and network security
- Portability and interoperability
- Client-server application design and development
- Sendmail and DNS
- GUI technologies and builders

To provide the best possible tutorial slate, USENIX is constantly soliciting proposals for new tutorials. If you are interested in presenting a tutorial, contact the Tutorial Coordinator: Daniel V. Klein, 412-421-2332, E-mail: *dvk@usenix.org*

TECHNICAL SESSIONS

The three days of technical sessions include one track of presentations of papers selected by the Program Committee. These refereed papers are published in the *Conference Proceedings* provided free to technical sessions attendees.

A parallel track of Invited Talks complements the refereed paper track within the technical sessions. These survey-style or tutorial-type talks by invited experts range over a variety of interesting topics, such as harnessing programming tools, resolving system administration difficulties, or employing specialized applications. *Submitted Notes* from the Invited Talks are published and distributed free to technical sessions attendees. This second track includes selections from the best presentations offered at recent technical symposia.

The Invited Talks Coordinators welcome suggestions for topics, as well as request proposals for particular Talks. In your proposal, state the main focus, include a brief outline, and be sure to emphasize why your topic is of general interest to our community. Please submit to: Brent Welch, Xerox PARC (1-415-812-4405) and Bob Gray, US WEST Advanced Technologies (1-303-541-6014) or via e-mail to *ITusenix@usenix.org*

WORK-IN-PROGRESS REPORTS

Work-In-Progress Reports (WIPs), scheduled during the technical sessions, introduce interesting new or ongoing work. The USENIX audience provides valuable discussion and feedback. Do you have interesting work you would like to share, or a cool idea that is not ready to be published? Work-In-Progress Reports are for you! We are particularly interested in presentations of student work. To schedule your report, contact Peg Schafer via e-mail to: *wips@usenix.org*

Birds-of-a-Feather Sessions

The always popular Birds-of-a-Feather (BoFs) sessions are offered Tuesday, Wednesday, and Thursday evenings of the conference. BoFs may be scheduled at the conference or in advance by telephone to the USENIX Conference Office at 1-714-588-8649 or via e-mail to *conference@usenix.org*

Vendor Displays

In the USENIX Vendor Display, emphasis is on serious answers from technically savvy vendor representatives. Here, in a relaxed environment, conference attendees can "kick the tires" and be sure the products and services on display really do what they're said to do. Plus, you can review the newest releases from technical publishers.

♦ **Vendors:** this is an exceptional opportunity for receiving feedback on new development from USENIX's technically astute conference attendees. If your company would like to demonstrate or display its products and services, please contact: Peter Mui, 1-510-528-8649, FAX: 1-510-548-5738, E-mail: *pmui@usenix.org*

Refereed Paper Submissions

As always at a USENIX conference, we are interested in papers describing new and interesting developments in and around open systems, including networking, new languages, tools, and software engineering. Papers that analyze problem areas and draw important conclusions from practical experience are encouraged.

Note that the USENIX conference, like most conferences and journals, requires that papers not be submitted simultaneously to more than one conference or publication and that submitted papers not be previously or subsequently published elsewhere.

♦ **Cash Prizes**

Cash prizes will be awarded for the best paper at the conference, the best paper by a full-time student, and the best presentation.

♦ **How to Submit**

It is important that you contact the USENIX Association office to receive detailed guidelines for submitting a paper; e-mail to *summer94authors@usenix.org* In addition, enquiries about submissions to the USENIX Summer 1994 Conference may be made by e-mail to *margo@usenix.org*

The schedule for reviewing submissions for the conference is very short, so our preferred submission is an extended abstract (although we will accept full papers). The object of an extended abstract is to convince the reviewers that a good paper and 25-minute presentation will result. They need to know that authors:

- are attacking a significant problem
- are familiar with the current literature about the problem
- have devised an original or clever solution
- if appropriate, have implemented the solution and characterized its performance
- have drawn appropriate conclusions about what they have learned

In brief, the extended abstract should be 5 manuscript pages (single-sided) or less in length. It should represent the paper in "short form." Include the abstract as it will appear in the final paper. Include references and, if appropriate, performance data to establish that you have a working implementation and measurement tools. If the full paper has been completed, it may be submitted instead of an extended abstract. Full papers should be limited to 12 single-spaced pages.

Every submission should include one additional page containing:

- the name of one of the authors, who will act as the contact for the program committee
- contact's surface mail address, daytime and evening telephone numbers, e-mail address, and (if available) fax number
- an indication of which, if any, of the authors are full-time students

Where to Send Submissions

Submit one copy of an extended abstract or full paper by January 28, 1994 via *at least two* of the following methods:

- E-mail to *summer94papers@usenix.org*
- FAX to +1-510-548-5738
- Mail to: Summer 1994 USENIX, USENIX Association, 2560 Ninth St., Ste. 215, Berkeley, CA U.S.A. 94710

Dates For Refereed Paper Submissions

- **Submissions due:** January 28, 1994
- **Authors notified by:** February 21, 1994
- **Camera-ready papers due:** April 8, 1994

Program Committee

- **Program Co-Chairs:** Margo Seltzer, *Harvard University* Keith Bostic, *University of California, Berkeley*

Mary Baker, *Stanford University*

John Bongiovanni, *Stratus Computer, Inc.*

Glen Dudek, *Object Design, Inc.*

Peter Honeyman, *University of Michigan*

Michael Jones, *Microsoft Research, Microsoft Corporation*

Kent Peacock, *Consultant*

Rob Pike, *AT&T Bell Laboratories*

Joseph Provino, *Sun Select, Inc.*

Greg Rose, *Australian Computing and Communications Institute*

Win Treese, *Massachusetts Institute of Technology*

Ozan Yigit, *York University*

For More Information

Materials containing details of the technical and tutorial program, conference registration, hotel and airline discount and reservation information will be mailed at the end of March 1994. If you wish to receive the preregistration materials, please contact:

- USENIX Conference Office,
  22672 Lambert St., Suite 613
  Lake Forest, CA U.S.A. 92630
  Phone: 1-714-588-8649
  FAX: 1-714-588-9706
  E-mail: *conference@usenix.org*

The UNIX and Advanced Computing Systems Professional and Technical Association

1994 USENIX Symposium on High-Speed Networking

AUGUST 1–2, 1994
CLAREMONT HOTEL & RESORT
OAKLAND, CALIFORNIA

Refereed Paper Submissions

- Extended abstracts due: May 2, 1994
- Notification to authors: May 16, 1994
- Camera-ready final papers due: June 20, 1994

Symposium Schedule

Registration Materials Available:
- June 1994

Monday, August 1
- Keynote address, followed by technical sessions
- Reception
- Birds-of-a-Feather sessions

Tuesday, August 2
- Technical sessions

Field Trip

Wednesday, August 3
Join us for visits to two high-speed networking testbeds, XUNET/BLANCA and CalREN, in Berkeley.

High-speed, high-capacity networks promise to change profoundly the way we compute. Fast, wide-area networks pose fresh challenges even for mature operating systems, such as UNIX. How will these innovations shape the design of future operating systems? Can we devise applications that fully (and productively) consume the bandwidth at our disposal?

The goals of this symposium are to encourage the UNIX and high-speed networking communities to commingle, to examine the issues and trends in high-speed networking, and to explore the impact of high-speed networks on systems and applications design.

The single-track symposium offers two days of technical presentations (followed by a field trip on the third day). Formally reviewed papers will be presented and published in the *Symposium Proceedings*. A copy of the *Proceedings* will be distributed to all attendees; additional copies may be purchased from the USENIX Association.

Symposium Topics

We seek presentations of original work on these (and related) topics:
- Network architectures
- Operating system support for high-speed networks
- Protocols
- Performance
- Network management
- Applications
- Practical experiences

Refereed Paper Submissions

If you are interested in presenting your work at the symposium, please submit an extended abstract as described below. The extended abstract should represent the final paper in "short form." Its object is to persuade the Program Committee that you will deliver a good 20–25 minute presentation and final paper.

The Committee needs to know that authors:
- are tackling a significant problem.
- are familiar with the current literature about the problem.
- have devised an original solution.
- have implemented the solution and, if appropriate, have characterized its performance.
- have drawn appropriate conclusions about what they have learned and why it is important.

Note that the Program Committee considers it unethical to submit the same paper simultaneously to more than one conference or publication, or to submit a paper that has been or will be published elsewhere, without disclosing this information with the submission.

If your paper is accepted, you are expected to provide a full paper in camera-ready form for publication in the *Proceedings* and to present your work at the Symposium.

How to Submit

A typical extended abstract is roughly 2500 words (5 pages). Indicate clearly whether the paper represents a design, an implementation or a system that is in wide use. You are encouraged to include references. Supporting material may be in note or outline form. If you wish, you may supplement the extended abstract with a copy of a full paper.

Please submit one copy of an extended abstract using *at least two* of the following methods:

- E-mail (*preferred method*) to: *net94papers@usenix.org*
- Mail to:
  Pat Parseghian, Program Chair
  AT&T Bell Laboratories, Room 2C-472
  600 Mountain Avenue
  PO Box 636
  Murray Hill NJ USA 07974-0636
- FAX to: Pat Parseghian +1 (908) 582-5857

Please, with your submission, include the following information about the author(s):

- Name (indicate which author will serve as the contact)
- Affiliation
- Daytime telephone
- Postal address
- E-mail address
- FAX number

For More Program Information

Refer questions about refereed paper submissions and other program concerns to the Program Chair:

- Pat Parseghian
  Telephone: +1 (908) 582-4229
  E-mail: *pep@research.att.com*

USENIX, the UNIX and Advanced Computing Systems Professional and Technical Association.

Dates For Refereed Paper Submissions

Extended abstracts due:
May 2, 1994

- Notification to authors:
  May 16, 1994
- Camera-ready final papers due: June 20, 1994

Program Committee

- **Program Chair:**
  Pat Parseghian,
  AT&T Bell Laboratories

Bill Johnston, *Lawrence Berkeley Laboratory*

Tom Lyon, *Sun Microsystems*

Jeffrey Mogul,
Digital Equipment Corporation, Western Research Laboratory

Gerald Neufeld, *University of British Columbia*

Lixia Zhang, *Xerox PARC*

For Registration Information

Materials containing full details of the symposium program, symposium registration fees and forms, and hotel discount and reservation information will be available early June 1994. If you wish to receive the registration materials, please contact:

- USENIX Conference Office
  22672 Lambert St, Suite 613
  Lake Forest, CA USA 92630
  Phone: +1 (714) 588-8649
  FAX: +1 (714) 588-9706
  E-mail:
  conference@usenix.org

ANNOUNCEMENT/CALL FOR PARTICIPATION

8th USENIX Systems Administration Conference (LISA VIII)

SEPTEMBER 19–23, 1994
TOWN AND COUNTRY HOTEL
SAN DIEGO, CALIFORNIA

Co-sponsored by SAGE, the System Administrators Guild

The annual USENIX Systems Administration Conference provides a forum in which system administrators meet to share ideas and experiences. A growing success for the previous seven years, the USENIX Systems Administration Conference is the only conference which focuses specifically on the needs of system administrators. Its scope includes system administrators from sites of all sizes and configurations.

"Automation: Managing the Computer of the 90's" is the theme of this year's conference. The conference will focus on tools to help system administrators automate administration tasks and troubleshoot problems.

Tutorial Program

♦ **Monday and Tuesday, September 19–20, 1994**

The two-day tutorial program at the conference offers multiple tracks, with a total of as many as twelve half-day tutorials. Attendees may move between tracks, choosing the sections of most interest to them. Tutorials offer expert instruction in areas of interest to system administrators, novice through experienced. Topics are expected to include Networking, Advanced System Administration Tools, Solaris & BSD Administration, Perl Programming, System Security, and more.

Technical Sessions

♦ **Wednesday through Friday, September 21–23, 1994**

The three days of technical sessions program will include refereed paper presentations, invited talks, panels, Works-In-Progress (WIP) reports, and Birds-Of-a-Feather (BOF) sessions. The first track is dedicated to presentations of referred technical papers. Although papers of a traditional technical content are very welcome, the Program Committee is especially seeking papers on areas such as useful tools or solutions to system administration problems. Papers which are tutorial in nature would also be appropriate. The second track of the Technical Sessions will offer invited talks, panels, mini-workshops, and similar presentations, and we seek proposals for these presentation formats as well.

Conference Proceedings, containing all refereed papers and materials from invited talks and workshops, will be distributed to conference attendees. The *Conference Proceedings* will also be available from the USENIX Association following the conference.

Vendor Display

♦ **Wednesday, September 21, 1994, 3:00 pm–9:00 pm**

Well informed vendor representatives will demonstrate products and services useful to systems and network administration at the informal table-top display accompanying the USENIX Systems Administration Conference. If your company would like to participate, please contact Peter Mui at (510) 528-8649; FAX (510) 548-8649; E-mail: *pmui@usenix.org*

Conference Topics

The Program Committee invites you to submit to the refereed paper track of the technical sessions, as well as submit informal proposals, ideas, or suggestions for the various presentation formats of the second track, on any of the following or related topics:

♦ Automating Administration Tasks
♦ Distributed System Administration

Important Dates

Refereed Paper Submissions:

♦ Extended Abstract Submission Deadline: May 23, 1994
♦ Notification to Authors: June 24, 1994
♦ Final Papers Receipt Deadline: August 1, 1994

Registration Materials Available:

♦ July, 1994

- Problem Tracking
- Predicting problems before they happen
- System Administration standards
- Differences in OSF, Solaris, and ?
- Case studies – "This is the problem we solved and how we solved it."
- Career paths for system administrators ("Is there life after support?")
- Applications using emerging technology (C++, AI, etc.)
- Performance Monitoring
- Hardware-related topics: all about memory, installing disk drives
- Tools – Useful programs or solutions you have developed and wish to share

Refereed Paper Submissions

We strongly urge you to request a sample extended abstract by sending e-mail to *sample-abstract@usenix.org* or telephoning +1 (510) 528-8649.

The Program Committee requires that an extended abstract be submitted for the paper selection process. (Full-papers are not acceptable for this stage; if you send a full paper, you must also include an extended abstract for evaluation.) Your extended abstract should consist of a traditional abstract which summarizes the content/ideas of the entire paper, followed by a skeletal outline of the full paper.

Submissions will be judged on the following criteria: relevancy of topic, quality of work, and quality of the written submission.

Note that the USENIX conference, like most conferences and journals, considers it unethical to submit the same paper simultaneously to more than one conference or publication or to submit a paper that has been or will be published elsewhere.

Authors of an accepted paper will present their paper at the conference and provide a final paper for publication in the *Conference Proceedings*. Final papers are limited to 20 pages, including diagrams, figures and appendix and must be in troff or ASCII format. We will supply you with instructions and troff macros. Papers should include a brief description of the site (if applicable).

Where to Send Submissions

For submission to the refereed paper track, please send submissions by *at least two* of the following methods:

- (*Preferred method*) electronic (nroff/troff or ASCII) submission of the extended abstract; e-mail to: *dinah@usenix.org*
- FAX to the USENIX Association +1 (510) 548-5738
- Mail to: LISA 8 Conference, USENIX Association, 2560 Ninth St., Suite 215, Berkeley, CA USA 94710

For submission of all proposals other than extended abstracts of refereed papers, and for inquiries regarding the content of the conference program, contact the Program Chair: Dinah McNutt, Tivoli Systems, P.O. Box 202253, Austin, TX USA 78720-2253, +1 (512) 267-9381, E-mail: *dinah@usenix.org*

USENIX, the UNIX and Advanced Computing Systems Professional and Technical Association.

Dates For Refereed Paper Submissions

- Extended Abstract Submission Deadline: May 23, 1994
- Notification to Authors: June 24, 1994
- Final Papers Receipt Deadline: August 1, 1994

Program Committee

- **Program Chair:** Dinah McNutt, *Tivoli Systems*

Tom Christiansen, *Consultant*

Trent Hein, *XOR Network Engineering*

William (Bill) LeFebvre, *Northwestern University*

Pat Parseghian, *AT&T Bell Laboratories*

Hal Stern, *Sun Microsystems*

Jeff Tate, *Bank of America*

Mark Verber, *Xerox PARC*

Neil Todd, *GID Ltd*

For Registration Information

Materials containing all details of the symposium program, symposium registration fees and forms, and hotel discount and reservation information will be mailed and posted to the net beginning July 1994. If you wish to receive registration materials, please contact:

- USENIX Conference Office
  22672 Lambert Street, Suite 613
  Lake Forest, CA USA 92630
  +1 (714) 588-8649
  FAX: +1 (714) 588-9706
  E-mail: *conference@usenix.org*

ANNOUNCEMENT & CALL FOR PARTICIPATION

VHLL

USENIX Symposium on Very High Level Languages

OCTOBER 26–28, 1994
EL DORADO HOTEL
SANTA FE, NEW MEXICO

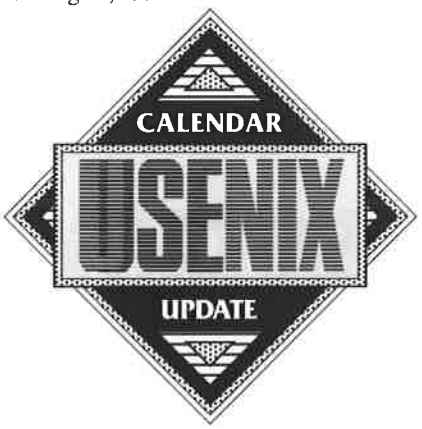
Important Dates

DATES FOR REFEREED PAPER SUBMISSIONS:

- Extended Abstracts Due: June 30, 1994
- Notifications to Authors: July 27, 1994
- Final Papers Due: Sept. 12, 1994

REGISTRATION MATERIALS AVAILABLE:

- August, 1994



Using very high level languages (VHLLs), programmers can assemble entire applications from large building blocks in just a small fraction of the time required if conventional programming strategies were used. These languages allow programmers to take advantage of increasingly available hardware cycles, trading cheap machine time for costly programmer time. Thus, VHLLs offer one of the most promising approaches toward radically improving programmer productivity.

UNIX has long supported very high level languages; consider *awk* and the various shells. Often programmers create what are essentially new little languages whenever a problem appears of sufficient complexity to merit a higher level programming interface – consider *sendmail.cf*. In recent years many UNIX programmers have been turning to VHLLs for both rapid prototypes and complete applications. They take advantage of these languages' higher level of abstraction to complete projects more rapidly and more easily than they could have using lower-level languages.

Some VHLLs such as *TCL*, *Perl*, *Icon*, and *REXX* have gained widespread use and popularity. Many others never see the public light. Some of these languages are special purpose, addressing a limited-problem domain (such as graphics, text processing, or mathematical modeling) using powerful primitives created for that specific problem. Other VHLLs are more general purpose in nature, but still much higher level than most traditional compiled languages. Some are stand-alone languages, while others are designed to be embedded in other programs. Many are interpreted, although some are compiled to native machine code; a few occupy a gap between both worlds.

Symposium Scope and Format

The USENIX Symposium on Very High Level Languages will spotlight these languages and their usefulness in leveraging certain kinds of tasks. The Symposium will introduce participants to concepts and approaches they haven't examined yet, and publish original work in these areas. Programmers will learn about the relative strengths and weaknesses and extract the key concepts that run through the various languages presented.

The USENIX Symposium on Very High Level Languages will run three days:

- Wednesday, October 26, will feature hour-long overviews by invited speakers of some of the more popular VHLLs in use today, such as *TCL*, *Perl*, *Icon*, and *REXX*.
- Thursday and Friday, October 27–28, will consist of refereed papers, tutorial-style invited talks on related topics, and panel discussions.
- Birds-of-a-Feather sessions will be held Wednesday and Thursday evenings, and a Reception will be held Thursday evening.

Papers on brand-new languages, on existing languages about which little or nothing has been published, on applications that use these languages in creative fashions not yet seen, and on experiences at extending existing languages (for example, adding windowing capabilities to *awk*) are all welcome. Papers should address designing, building, testing, debugging, and measuring the performance and usability of these languages, as well as reference and compare related work in the area. Mention both advantages and disadvantages of the approach selected. For applications using these languages, compare and contrast the design, development, and support effort that were required with this approach versus one using a lower-level language. Good papers will be of interest to people who use other VHLLs than the one described in the paper. For example, a paper describing a system built in a particular language will be much more interesting if it highlights some important feature of the language or problems with the language, or some issue relevant to VHLLs in general.

(conitnued on reverse side)

Program Committee

♦ **PROGRAM CHAIR:**
Tom Christiansen, *Consultant*

Stephen C. Johnson,
Melismatic Software

Brian Kernighan,
AT&T Bell Laboratories

John Ousterhout,
University of California, Berkeley

Henry Spencer,
University of Toronto

Dates For Refereed Paper Submissions

- ♦ Extended Abstracts Due: June 30, 1994
- ♦ Notifications to Authors: July 27, 1994
- ♦ Final Papers Due: September 12, 1994

USENIX

How to Submit to the Symposium

Persons interested in participating in panel discussions or organizing Birds-of-a-Feather sessions should contact the program chair as indicated below.

Submissions of papers to be presented at the Symposium and published in the *Symposium Proceedings* must be in the form of an extended abstract. The extended abstract should be 1500–2500 words (3–5 pages) and must be received by June 30, 1994. (If you do send a full paper, you must also include an extended abstract for evaluation.) The extended abstract should represent your paper in short form. Its purpose is to convince the program committee that a good paper and presentation will result. You should show that you are addressing an interesting problem, have surveyed existing solutions, have devised an innovative, and original solution, and have drawn appropriate conclusions about what has been learned.

All submissions should indicate the electronic mail address and telephone number of a principal contact. Authors will be notified of acceptance by July 27, 1994, and will be provided with guidelines for preparing camera-ready copy of the final paper. The final paper must be received no later than September 12, 1994. Note that the USENIX conference, like most conferences and journals, considers it unethical to submit the same paper simultaneously to more than one conference or publication or to submit a paper that has been or will be published elsewhere.

Please submit your extended abstracts to the program chair as follows:

EMAILED SUBMISSIONS (PREFERRED):

- ♦ must be in ASCII, troff (with the -me macro set or raw troff preferred), or Postscript form
- ♦ send to tchrist@usenix.org

HARD COPY SUBMISSIONS:

- ♦ via FAX to +1 (303) 442-7177 (Please refer to Tom Christiansen)
- ♦ via postal mail, please submit 6 paper copies to:

  Tom Christiansen
  USENIX VHLL Symposium
  2227 Canyon Blvd., #262
  Boulder, CO 80302

For Program and Registration Information

Materials containing full details of the symposium program, registration fees and forms, and hotel discount and reservation information will be mailed and posted to the net in August 1994. If you wish to receive these materials, please contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA USA 92630
+1 (714) 588-8649; FAX: +1 (714) 588-9706
Internet: conference@usenix.org

USENIX, the UNIX and Advanced Computing Systems Professional and Technical Association.

USENIX Online Library and Index

What Is It

The USENIX online index is an electronically available list of papers published by the USENIX Association and related groups. The index is kept as a simple ASCII file, in refer/bib format, sorted by author, and contains information about papers published in USENIX conference and workshop proceedings, newsletters, journal, and the like.

The index is updated approximately monthly.

How to Get the Index

The index is available online from UUNET, either via a mail server or anonymous ftp. The index is about 200K, and available only in its entirety.

To get it as mail, send mail to *library@uunet.uu.net* with "send bibliography" as the contents of your message.

To get it via ftp from ftp.uu.net, login as "anonymous" with your email address as the password. Then:

```
ftp> cd library
250 CWD command successful.
ftp> get bibliography
```

This help file can be retrieved with "send help" or as the ftp file "help.bibliography". (There is no person associated with the library address and it will never be read by human eyes.)

How to Access Information

To build the indices so you can easily access information, run "indxbib" on the bibliography: *indxbib filename*. You can then pull information from the file by running "lookbib". You can either build refer files or run lookbib interactively.

For example, the following command would put all entries which refer to Smith into a file called "stuff":

```
echo smith | lookbib bibliography > stuff.
```

Or you could interact with the index by saying:
`lookbib bibliography`

It will ask you if you want instructions when it starts, answer yes. Then at the prompt, for example: `>smith`

will list references to smith (upper/lower case doesn't matter).

To Get an Online Paper

As of this date, we have not yet set up the online papers. When this capability is provided, we will announce it on the net and these instructions will be updated with retrieval information.

Publications Indexed

USENIX:
Conference proceedings, workshop and symposia proceedings, *Computing Systems* journal, newsletter

EurOpen (formerly EUUG - European UNIX Users Group):
Conference proceedings, newsletter (1982-1989)

Other sources are being continually evaluated and will be included as deemed suitable.

Fields Used in the Index

The standard bib/refer formats are used. These include:

A	Author (may be multiple entries)
T	Title of article
P	Page number(s)
W	Primary author's institution
I	Issuer/publisher
B	Conference proceedings or book title
J	Name of newsletter or journal
D	Date of publication or conference
C	Location of conference
V	Volume number
N	Nbr within volume
O	Other comments (e.g., "Abstract only")

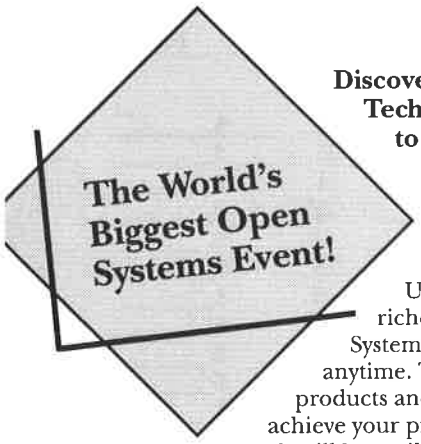
These fields may be extended to include other information such as identifier for retrieval, keywords, online format of paper (PostScript, troff, etc.), language (if other than English), etc.

More Information

For additional information about the online index and library, and/or instructions for donating papers, contact: *index@usenix.org*

Or write to:
USENIX Association
2560 Ninth St., Suite 215
Berkeley CA 94710

Open Systems is Your Destination. UniForum Gets You There First.



Discover New Products, New Technologies and New Ideas to help you face the challenges of an Open Systems future.

Over 400 leading companies will make UniForum '94 one of the richest IS resources for Open Systems professionals anywhere, anytime. Thousands of essential products and services that can help you achieve your productivity and profitability goals will be available for you to evaluate.

Special Exhibit Events and Demonstrations include:

- UniForum World of the Future Pavilion
- DCE Demonstration
- New Product Award Competition
- Access to the Data Highway
- Virtual Reality
- And more

All New 5 Day Conference Program*

Designed to meet the needs of both the UNIX and Open Systems Veteran and the IS Professional from proprietary environments, UniForum's conference program has the answers.

Monday & Tuesday, March 21-22 Special 2-day programs

- Second Annual Technology Managers' Conference From Pilot to Production: Implementing Client/Server Applications in The Real World
- USENIX/SAGE Systems Administration Tutorials For Systems Managers and Administrators

Tutorials - Intensive One Day Sessions

- Performance Measurement/Management and Capacity Planning of UNIX Systems
- Basics of Object-Oriented Technology
- TCP/IP: Understanding the Protocols of the Internet
- Enterprise Messaging
- Distributed Systems Management
- Internet For Beginners
- Multimedia Systems: A Guided Tour
- Internetwork Management In Transition: Moving From SNMP Version 1 to SNMP Version 2
- Introduction to ATM
- Mastering Fundamental UNIX

*As of 10/7/93 Subject to Change

Plenary Session - FREE to All Attendees

Wednesday, March 23
Politics of Open Architectures: Sincerely Expanding Open Systems or Manipulating the Market for Acceptance

Thursday, March 24
Windows NT: UNIX Threat or Paper Tiger?

Friday, March 25
Comparison of Hardware Vendors Open Systems Strategies

Workshops - 1/2 Day Hands-On Classroom Sessions

Friday, March 25

- Exchanging E-mail Between Two Systems
- Synchronizing E-mail Directories Between or Among Two or More Systems
- Exchanging Files Between Two Systems
- Properly Executing an Application Running on Another System

Seven Conference Tracks Cover The Issues

Wednesday thru Friday, March 23-25

- Global Networks And The Role of UNIX
- A New Age For Enterprise System Strategies
- Open Systems:Truths and Myths
- Costs And Benefits – A Manager's Perspective On UNIX
- UNIX And Large Scale Data Management
- Trends In Software Development
- UniForum Opens The Future

For more information call toll free (800) 225-4698 (in U.S.) or (508) 879-6700. For fastest service, fill in the coupon below and fax to: (508)872-8237.

UNIFORUM 1994
MARCH 21-25 • MOSCONE CENTER • SAN FRANCISCO
Your Roadmap to an Open Future

Fax to (508) 872-8237 or mail to: UniForum '94 c/o IDG, World Expo, P.O. Box 9107, Framingham, MA 01701- 9107

Name _____
Title _____
Company _____
Address _____
City _____ State _____
Zip _____
Code USE

UniForum'94 is sponsored by UniForum, the International Association of Open Systems Professionals.

Publications Order Form

Conference & Workshop Proceedings

Qty	Proceedings		Member Price	Non-Member Price	Subtotal	Overseas Postage	Total
WINTER/SUMMER CONFERENCES							
___	Cincinnati	Summer '93	25	33	$_____	$18	$_____
___	San Diego	Winter '93	33	40	$_____	25	$_____
___	San Antonio	Summer '92	23	30	$_____	14	$_____
___	San Francisco	Winter '92	30	39	$_____	22	$_____
___	Nashville	Summer '91	32	38	$_____	22	$_____
___	Dallas	Winter '91	28	34	$_____	18	$_____
___	Anaheim	Summer '90	22	22	$_____	15	$_____
___	Washington, DC	Winter '90	25	25	$_____	15	$_____
___	Baltimore	Summer '89	20	20	$_____	15	$_____
___	San Diego	Winter '89	30	30	$_____	20	$_____
___	San Francisco	Summer '88	29	29	$_____	20	$_____
___	Dallas	Winter '88	26	26	$_____	15	$_____
___	Phoenix	Summer '87	35	35	$_____	20	$_____
___	Washington, DC	Winter '87	10	10	$_____	15	$_____
___	Atlanta	Summer '86	37	37	$_____	20	$_____
___	Denver	Winter '86	25	25	$_____	15	$_____
___	Portland	Summer '85	45	45	$_____	25	$_____
___	Dallas	Winter '85	15	15	$_____	10	$_____
___	Salt Lake City	Summer '84	29	29	$_____	20	$_____
___	Washington, DC	Winter '84	25	25	$_____	15	$_____
___	Toronto	Summer '83	32	32	$_____	20	$_____
___	San Diego	Winter '83	28	28	$_____	15	$_____
LARGE INSTALLATION SYSTEMS ADMINISTRATION							
___	LISA VII	Nov. '93	25	33	$_____	12	$_____
___	LISA VI	Oct. '92	23	30	$_____	12	$_____
___	LISA V	Sept. '91	20	23	$_____	11	$_____
___	LISA IV	Oct. '90	15	18	$_____	8	$_____
___	LISA III	Sept. '89	13	13	$_____	9	$_____
___	LISA II	Nov. '88	8	8	$_____	5	$_____
___	LISA I	April '87	4	4	$_____	5	$_____
C++							
___	C++ Conference	Aug. '92	30	39	$_____	20	$_____
___	C++ Conference	April '91	22	26	$_____	11	$_____
___	C++ Conference	April '90	28	28	$_____	18	$_____
___	C++ Conference	Oct. '88	30	30	$_____	20	$_____
___	C++ Workshop	Nov. '87	30	30	$_____	20	$_____
SECURITY							
___	UNIX Security IV	Oct. '93	15	20	$_____	9	$_____
___	UNIX Security III	Sept. '92	30	39	$_____	11	$_____
___	UNIX Security II	Aug. '90	13	16	$_____	8	$_____
___	UNIX Security	Aug. '88	7	7	$_____	5	$_____
MACH							
___	Mach Symposium III	April '93	30	39	$_____	18	$_____
___	Mach Symposium	Nov. '91	24	28	$_____	14	$_____
___	Mach Workshop	Oct. '90	17	20	$_____	9	$_____

Qty	Proceedings		Member Price	Non-Member Price	Subtotal	Overseas Postage	Total
DISTRIBUTED & MULTIPROCESSOR SYSTEMS (SEDMS)							
___	SEDMS IV	Sept. '93	24	32	$_____	14	$_____
___	SEDMS III	Mar. '92	30	36	$_____	20	$_____
___	SEDMS II	Mar. '91	30	36	$_____	20	$_____
___	SEDMS	Oct. '89	30	30	$_____	20	$_____
MICROKERNELS & OTHER KERNEL ARCH.							
___	Microkernels & Other Kernel... II	Sept. '93	15	20	$_____	9	$_____
___	Microkernels & Other Kernel... I	Apr. '92	30	39	$_____	20	$_____
GRAPHICS							
___	Graphics Workshop V	Nov. '89	18	18	$_____	10	$_____
___	Graphics IV	Oct. '87	10	10	$_____	10	$_____
___	Graphics III	Nov. '86	10	10	$_____	5	$_____
___	Graphics II	Dec. '85	7	7	$_____	5	$_____
OTHER WORKSHOPS							
___	Mobile Computing	Aug. '93	15	20	$_____	8	$_____
___	File Systems	May '92	15	20	$_____	9	$_____
___	UNIX Transaction Processing	May '89	12	12	$_____	8	$_____
___	Software Management	Apr. '89	20	20	$_____	15	$_____
___	UNIX & Supercomputers	Sept. '88	20	20	$_____	10	$_____

Discounts are available for bulk orders. Please inquire.

Total price of Proceedings _____ **
Calif. residents add sales tax _____
Total overseas postage _____
Total enclosed _____

****If you are paying member price, please include member's name and/or membership number**_____

PAYMENT OPTIONS*

___ Check enclosed - payable to USENIX Association

___ Charge my: ___ VISA ___ MC Account # _____ Exp.Date _____

___ Purchase order enclosed Signature _____

* Outside the USA? Please make your payment in US currency by one of the following:
- Check - issued by a local branch of a US Bank
- Charge (VISA, MasterCard, or foreign equivalent)
- International postal money order

Shipping Information

Please allow 2–3 weeks for delivery. Overseas orders are shipped via air printed matter.

Ship to: _____

Please mail or fax this order form with payment to:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
FAX 510/548-5738

• If you are not a member and wish to receive our membership information packet, please check this box. ☐

Local User Groups

The Association will support local user groups by doing a mailing to assist in the formation of a new group and publishing information on local groups in *;login:*. At least one member of the group must be a current member of the Association. Send additions and corrections to: <*login@usenix.org*>.

CA – Fresno:

The Central California UNIX Users Group consists of a uucp-based electronic mailing list to which members may post questions or information. For connection information:

Educational and governmental institutions:
Brent Auernheimer (209) 278-2573,
<*brent@CSUFresno.edu* or *csufres!brent*>

Commercial institutions or individuals:
Gordon Crumal (209) 251-2648
<*csufres!gordon*>

CA – Orange County:

Meets the 2nd Monday of each month

UNIX Users Association of Southern California
Paul Muldoon (714) 556-1220 ext. 137
New Horizons Computer Learning Center
1231 E. Dyer Rd., Suite 140
Santa Ana, CA 92705

CO – Boulder:

Meets monthly at different sites. For meeting schedule, send email to < *fruug-info@fruug.org*>.

Front Range UNIX Users Group
Software Design & Analysis, Inc.
1113 Spruce St., Ste. 500
Boulder, CO 80302
Steve Gaede (303) 444-9100
<*gaede@fruug.org*>

D.C.– Washington, D.C.:

Meets 1st Tuesday of each month.

Washington Area UNIX Users Group
9811 Mallard Drive
Laurel, MD 20708
Alan Fedder (301) 953-3626

FL – Coral Springs

S. Shaw McQuinn (305) 344-8686
8557 W. Sample Road
Coral Springs, FL 33065

FL - Melbourne:

Meets the 3rd Monday of every month.

Space Coast UNIX User's Group
Steve Lindsey (407) 242-4766
<*lindsey@vnet.ibm.com*>

FL – Orlando:

Meets the 3rd Thursday of each month.

Central Florida UNIX Users Group
Mikel Manitius (407) 444-8448
<*mikel@aaa.com*>

FL – Western:

Meets 1st Thursday of each month.

Florida West Coast UNIX Users Group
Richard Martino (813) 536-1776
Tony Becker (813) 799-1836
<*mcrsys!tony*>
Ed Gallizzi, Ph.D. (813) 864-8272
<*e.gallizzi@compmail.com*>
Jay Ts (813) 979-9169
<*uunet!pdn!tscs!metran!jan*>
Dave Lewis (407)242-4372
<*dhl@ccd.harris.com*>

GA – Atlanta:

Meets on the 1st Monday of each month in White Hall, Emory University.

Atlanta UNIX Users Group
P.O. Box 12241
Atlanta, GA 30355-2241
Mark Landry (404) 365-8108

KS or MO – Kansas:

Meets on 2nd Monday of each month.

Kansas City UNIX Users Group (KUUG)
813B Street
Blue Springs, MO 64015
(816) 235-5212
<*mlg@cstp.umkc.edu*>

MI – Detroit/Ann Arbor

Meets on the 2nd Thursday of each month in Ann Arbor.

Southeastern Michigan Sun Local Users Group and Nameless UNIX Users Group
Steve Simmons office: (313)769-4086
home: (313) 426-8981
<*scs@lokkur.dexter.mi.us*>

MN – Minneapolis/St. Paul:

Meets the 1st Wednesday of each month.

UNIX Users of Minnesota
17130 Jordan Court
Lakeville, MN 55044
Robert A. Monio (612) 220-2427
<pnessutt@dmshq.mn.org>

MO – St. Louis:

St. Louis UNIX Users Group
P.O. Box 2182
St. Louis, MO 63158
Terry Linhardt (314) 772-4762
<uunet!jgaltstl!terry>

NE– Omaha:

Meets monthly.

/usr/group/nebraska
P.O. Box 31012
Omaha, NE 68132
Phillip Allendorfer (402) 423-1400

New England – Northern:

Meets monthly at different sites.

Peter Schmitt 603) 646-2085
Kiewit Computation Center
Dartmouth College
Hanover, NH 03755
<peter.schmitt@dartmouth.edu>

NJ – Princeton:

Meets monthly.

Princeton UNIX Users Group
Mercer County Community College
1200 Old Trenton Road
Trenton, NJ 08690
Peter J. Holsberg (609) 586-4800
<mccc!pjh>

NM – Albuquerque:

ASIGUNIX meets every 3rd Wednesday of each month. Phil Hortz 505/275-0466.

NY – New York City:

Meets every other month in Manhattan.

Unigroup of New York City
G.P.O. Box 1931
New York, NY 10116
<unigroup@murphy.com>
Bob Young
(212) 490-8470

OK - Tulsa:

Meets 2nd Wednesday of each month.

Tulsa UNIX Users Group, $USR
Stan Mason (918) 560-5329
<tulsix!smason@drd.com>
Mark Lawrence (918) 743-3013
<mark@drd.com>

TX – Austin:

Meets 3rd Thursday of each month.

Capital Area Central Texas UNIX Society
P.O. Box 9786
Austin, TX 78766-9786
<officers@cactus.org>
Tom Painter (512) 835-5457
<president@cactus.org>

TX – Dallas/Fort Worth:

Meets the 1st Thursday of each month.

Dallas/Fort Worth UNIX Users Group
P.O. Box 867405
Plano, TX 75086
Evan Brown (214) 519-3577
<evbrown@dsccc.com>

TX – Houston:

Meets 3rd Tuesday of each month.

Houston UNIX Users Group
(Hounix) answering machine (713) 684-6590
Bob Marcum, President (713) 270-8124
Chuck Bentley, Vice-president
(713) 789-8928
<chuckb@hounix.uucp>

WA – Seattle:

Meets monthly.

Seattle UNIX Group Membership Info.
Bill Campbell (206) 947-5591
6641 East Mercer
Mercer Island, WA 98040-0820
<bill@celestial.com>

CANADA – Toronto:

143 Baronwood Court
Brampton, Ont. Canada L6V 3H8
Evan Leibovitch (416) 452-0504
<evan@telly.on.ca>

CANADA – Ottawa:

The Ottawa Carleton UNIX Users Group
D.J. Blackwood (613)957-9305
<dave@revcan.rct.ca>

LISA Groups

Back Bay LISA (BBLISA)

New England forum covering all aspects of system and network administration, for large and small installations. Meets monthly, at MIT in Cambridge, MA.

For information, contact:

J.R. Oldroyd
Open Advisors Limited
(617) 227-5635
<*jr@opal.com*>

Mailing list subscription
requests: bblisa-request@cs.umb.edu
Mailing list postings: *bblisa@cs.umb.edu*

For current calendar of events,
finger bblisa@cs.umb.edu

BAY LISA

The Bay-LISA group meets monthly in Santa Clara, CA to discuss topics of interest for administration of sites with more than 100 users and/or computers.

Send email to <*baylisa-info@sysadmin.com*>
or contact:

Bjorn Satdeva
408/241-3111
<*bjorn@sysadmin.com*>

Statement of Ownership
Management and Circulation, 9/30/93

Title: *;login:*, Pub. No. 008334. Frequency: Bimonthly. Six issues published annually. Subscription price: $50 individuals and institutions. Location of office of publication: 2560 Ninth Street, Suite 215, Berkeley, Alameda County, CA, 94710. Headquarters of publishers: Same. Publisher: USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710. Editor: Ellie Young, 2560 Ninth Street, Suite 215, Berkeley, CA 94710. Owner: USENIX Association. The purpose and function, and nonprofit status of the organization and the exempt status for Federal income tax purposes have not changed during the preceding 12 months.

Extent and nature of circulation	Average no. copies each issue preceding 12 mos.	Actual no. copies of single issue published nearest to filing date
A.Total no. copies	5692	5700
B. Paid circulation, Mail subs.	5217	5076
C. Total paid circulation	5217	5076
D. Free distribution	267	350
E. Total distribution	5487	5426
F. Copies not distributed	208	274
G. Total	5692	5700

I certify that the statements made by me above are correct and complete.
Ellie Young, Executive Director

Calendar of Events

1994

Jan 10-14 IEEE 1003, Irvine, CA
17-21 * USENIX, San Francisco, CA
Mar 14-16 Interex ICMS, New Orleans, LA
21-25 * UniForum, San Francisco, CA
Apr 5- 9 * SANS III, Arlington, VA
11 -14 * C++, Cambridge, MA
18 -22 IEEE 1003, Lake Tahoe, CA
25-28 * UNIX Applications Development Toronto, Canada
May 2- 6 Networld+INTEROP 94, Las Vegas, NV
7-13 DECUS, New Orleans, LA
Jun 6-10 * USENIX, Boston, MA
" Networld+INTEROP 94, Berlin
Jul 11-15 IEEE 1003
Aug 1- 2 * High–Speed Networking, Berkeley, CA
Sept 12-14 Networld+INTEROP 94, Atlanta, GA
18 -22 Interex 94, Denver, CO
19 -23* LISA VIII, San Diego, CA
Oct 23-27 ACM OOPSLA, Portland, OR
26-28 * Very High Level Languages Santa Fe, NM
Nov 12-18 DECUS, Anaheim, CA
Fall * Symposium on Operating Systems

1995

Jan 16-20 * USENIX, New Orleans, LA
Feb 21-23 UniForum, Dallas, TX
May 13-19 DECUS, Atlanta, GA
Jun 19-23 * USENIX, San Francisco, CA
Aug 13-17 Interex 95, Toronto, Canada
Nov 2- 8 DECUS, San Francisco, CA

1996

Jan 22-26 * USENIX, San Diego, CA
Mar 12-14 UniForum, San Francisco, CA
May 18-24 DECUS, Orlando, FL
June 17-21* Washington, DC
Aug 4 - 8 Interex 96, San Diego, CA
Nov 16-22 DECUS, Anaheim, CA

This is a combined calendar of planned conferences, symposia, and standards meetings related to the UNIX operating system. If you have a UNIX-related event that you wish to publicize, please contact <*login@usenix.org*>. Please provide your information in the same format as above.

* = events sponsored by the USENIX Association.

ACM: Association for Computing Machinery
AUUG: Australian UNIX Users Group
DECUS: Digital Equipment Computer Users Society
EurOpen: European Forum for Open Systems

FedUNIX: Council of Advanced Computing Systems Technologists in Government
FIRST: Forum of Incidence Response & Security Team
GURU: Roumanian UNIX User Group
IEEE: Institute of Electrical and Electronics Engineers

IETF: Internet Engineering Task Force
INET: Internet Society
Interex: Intl. Association of Hewlett-Packard Comp.Users
JUS: Japan UNIX Society

LISA: USENIX Systems Administration Conference
NOSSDAV: Network and Operating System Support for Digital Audio and Video
OOPSLA: Object - oriented Programming Systems, Languages, and Applications
SAGE: System Administrators' Guild

SANS: System Administration, Networking & Security
UKUUG: United Kingdom UNIX Systems Users Group
UniForum: International Association of UNIX and Open Systems Professionals

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

Second Class Postage
PAID
At Berkeley, California
and Additional Offices

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710